Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implantação de Infraestrutura como Serviço em uma Nuvem Computacional Privada

Rafael César Merlo dos Santos

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia da Computação

Coordenador: Prof. Dr. Díbio Leandro Borges

Banca examinadora composta por:

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo (Orientadora) — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB
M.e Edward de Oliveira Ribeiro — Senado Federal

CIP — Catalogação Internacional na Publicação

dos Santos, Rafael César Merlo.

Implantação de Infraestrutura como Serviço em uma Nuvem Computacional Privada / Rafael César Merlo dos Santos. Brasília : UnB, 2016. 76 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. Computação em Nuvem, 2. Nuvem de Infraestrutura,
3. Plataformas de Infraestrutura como Serviço, 4. CloudStack,
5. Nuvem Privada

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implantação de Infraestrutura como Serviço em uma Nuvem Computacional Privada

Rafael César Merlo dos Santos

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo (Orientadora)
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
CIC/UnB

M.e Edward de Oliveira Ribeiro
Senado Federal

Prof. Dr. Díbio Leandro Borges
Coordenador do Curso de Engenharia da Computação

Brasília, 05 de agosto de 2016

Dedicatória

Dedico este trabalho a minha família que, desde sempre, não mediram esforços para que eu chegasse até esta etapa da minha vida, aos meus amigos por toda diversão e companheirismo e a minha namorada, Ana Gabriela, por todo seu amor e pelas alegrias compartilhadas.

Agradecimentos

Gostaria de agradecer a Universidade de Brasília (UnB), e todos os professores e profissionais que fizeram parte do meu processo de ensino, também, gostaria de agradecer à minha família pelo suporte e amor proporcionados para a realização deste trabalho.

Agradecimentos, em especial, à minha orientadora Aletéia Patrícia, por toda sua dedicação e motivação, ao Tiago Alves por toda sua disposição em sempre ajudar a solucionar problemas deste trabalho, e a todo o grupo de alunos do LABID (Laboratório de Bioinformática e Dados) pela dicas e ajuda.

Resumo

O aumento da complexidade de aplicações e sistemas resulta em uma procura maior por recursos computacionais (poder de processamento, espaço de armazenamento, etc.). Consequentemente, novos paradigmas e conceitos de computação surgem, e a computação em nuvem é um deles. Esse novo conceito visa oferecer serviços sobre demanda pela Internet, as chamadas nuvens computacionais. Os serviços podem ser categorizados em três modelos: infraestrutura, plataforma e software como serviço.

Este trabalho tem como foco o contexto de nuvens privadas que fornecem infraestrutura como serviço. Para isso, foi realizado um estudo sobre as principais plataformas de implantação para nuvens de infraestrutura. A proposta foi a implantação de uma nuvem privada de infraestrutura no laboratório LABID do departamento de Ciências da Computação da Universidade de Brasília. A plataforma CloudStack foi utilizada para a implantação da nuvem e, depois, foi avaliado o funcionamento e o comportamento através de testes funcionais e de desempenho. Este trabalho foi importante para garantir a infraestrutura interna de nuvem para os alunos e os pesquisadores do LABID, pois proporcionou a implantação de um ambiente de nuvem privada, que foi integrado a um ambiente de federação de nuvens pela plataforma BioNimbuZ.

Palavras-chave: Computação em Nuvem, Nuvem de Infraestrutura, Plataformas de Infraestrutura como Serviço, CloudStack, Nuvem Privada

Abstract

The increase in the complexity of applications and systems results in greater demand for computing resources (processing power, storage, etc.). Consequently, new computing concepts and paradigms arise and cloud computing is one of them. This new concept aims to offer services on-demand via the internet and can be categorized into three models: infrastructure-, platform- and software-as-a-service.

This work focuses on the context of private clouds providing infrastructure-as-a-service. A study of the major platforms for infrastructure cloud services was carried out, with the proposal of implementing a private infrastructure cloud in the LABID laboratory of the Computer Science Department of the University of Brasília. The CloudStack framework was used to implement the cloud and the operation and behaviour of the system were evaluated through functional and performance tests. This paper was of particular importance to LABID's students and researchers since it led to the implementation of an internal cloud environment which was then integrated into a cloud federation by the BioNimbuZ platform.

Keywords: Cloud Computing, Cloud Infrastructure, Frameworks for Cloud Infrastructure, Private Cloud

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A busca crescente por poder de processamento tem desenvolvido novos estudos e paradigmas na computação. Sistemas distribuídos como *clusters*, grades computacionais, e, mais atualmente, nuvens computacionais tem surgido como alternativa para suprir essa necessidade de maneiras distintas.

A computação em nuvem pode ser considerada um dos mais novos paradigmas de sistemas distribuídos. O termo se popularizou quando Eric Schmidt CEO (*Chief Executive Officer*) da Google o utilizou em uma conferência de *Search Engine* em 2006 para descrever um novo modelo de prestação de serviço através da Internet [11].

Analogamente, alguns serviços básicos, como, por exemplo, o fornecimento de água ou de eletricidade, permitem aos seus usuários receber tais serviços em praticamente qualquer lugar, e a qualquer hora. Assim, a computação em nuvem visa fornecer serviços de maneira transparente e sobre demanda para usuários interessados.

Para regular e definir melhor o assunto, de computação em nuvem, várias características como atendimento sobre demanda, elasticidade, medição de serviço, etc. são essenciais para a computação em nuvem. Além disso, há certos modelos de serviços de nuvem que visam fornecer infraestrutura (poder computacional, armazenamento e rede), plataforma (permite desenvolver aplicações de maneira mais flexível) e software (aplicações prontas para o uso que são executadas remotamente).

Também há modelos de implantação definidos para qualificar nuvens de computadores que são [12]: público, privado, comunitário e híbrido. Esses modelos estão relacionados em como os modelos de serviços são ofertados, por exemplo, provedores que desejam vender seus serviços podem utilizar o modelo público de implantação, ou se uma organização deseja que os serviços sejam de uso exclusivo da organização podem utilizar o modelo privado de implantação.

Um campo interdisciplinar que usufrui da computação em nuvem é a Bioinformática, pois trata de grandes quantidades de dados, e pode se beneficiar com os serviços ofertados

pelas nuvens computacionais. Com isso, diversas ferramentas de Bioinformática foram projetadas e implementadas desfrutando dos recursos disponibilizados pela computação em nuvens. O BioNimbuZ [13] é um projeto desenvolvido na UnB que utiliza a computação em nuvem para executar *workflows* de Bioinformática por meio nuvens que fornecem infraestrutura como serviço. O BioNimbuZ está em um cenário de federação de nuvens, o qual compreende o uso de diversos provedores de infraestrutura para fornecer seus serviços de Bioinformática.

Neste contexto, este trabalho propõe um estudo e a implantação de uma nuvem privada que forneça infraestrutura como serviço para o Laboratório de Bioinformática e Dados (LABID)[1], laboratório de pesquisa do departamento de Ciências da Computação da UnB, para suportar projetos de pesquisa como o BioNimbuZ.

1.1 Motivação

Este projeto tem a motivação de estudar a computação em nuvem e disponibilizar uma nuvem privada de infraestrutura para suportar projetos de bioinformática desenvolvido no LABID, como o BioNimbuZ, que utiliza a federação nuvens de infraestrutura para poder executar *workflows* de Bioinformática.

1.2 Problemas

Atualmente não exite na UnB, nenhuma nuvem de infraestrutura privada, ou seja, de controle da própia universidade para apoiar projetos que possam utilizar esses tipo de implantação e serviço. Como, por exemplo, o projeto BioNimbuZ é um projeto de Bioinformática desenvolvido no LABID e está em um cenário de federação de nuvens de infraestrutura.

1.3 Objetivos

O objetivo geral deste trabalho é implantar uma nuvem computacional privada de infraestrutura para o LABID. Para cumprir o objetivo geral deste trabalho faz-se necessário nalisar as principais tecnologias existentes em computação em nuvem, principalmente, das tecnologias para nuvens de infraestrutura e atingir os seguintes objetivos específicos:

1. Criar um modelo de implantação de uma nuvem privada de infraestrutura como serviço para o LABID;

[1] `http://www.cic.unb.br/pesquisa/laboratorios/`

2. Montar uma infraestrutura física com o modelo proposto;

3. Avaliar o comportamento da nuvem privada de infraestrutura implantada através de testes funcionais, de integração e de desempenho;

1.4 Organização do Trabalho

Este trabalho foi dividido em mais cinco capítulos, os quais são descritos a seguir: no Capítulo 2 são apresentados conceitos da computação distribuída, como *Clusters* e Grades computacionais, mas com o foco principal na computação em nuvem, uma vez que o objetivo principal deste trabalho é a implantação de uma nuvem privada de infraestrutura.

No Capítulo 3 será abordado o conceito de virtualização de hardware, essencial para nuvens de infraestrutura, e apresenta as principais plataformas para a implantação e o gerenciamento de nuvens privadas que oferecem infraestrutura como serviço.

O Capítulo 4 detalha a implantação da nuvem privada de infraestrutura no LABID, utilizando a plataforma CloudStack [8], a qual mais se destacou no estudo realizado no Capítulo 3. Este capítulo abordará o cenário em que a nuvem está inserida, a arquitetura definida para a implantação e as instalações e as configurações realizadas para a implantação.

No Capítulo 5 são apresentados os testes realizados e os resusltados obtidos com a implantação da nuvem privada para o LABID.

Por fim, o Capítulo 6 apresenta as conclusões obtidas e demais trabalhos futuros, que poderão otimizar a proposta do trabalho.

Capítulo 2

Computação em Nuvem

O propósito deste capítulo é expor conceitos que envolvem o paradigma de computação distribuída. O foco principal será em computação em nuvem, uma vez que o objetivo central deste trabalho é a implantação de uma nuvem privada. Para isso, serão abordados na Seção 2.1 conceitos da computação distribuída e os paradigmas de *clusters* e grades de computadores. E logo após, na Seção 2.2, detalhar os conceitos, as características e as particularidades que envolvem a computação em nuvem.

2.1 Sistemas Distribuídos

Tanenbaum [1] define sistemas distribuídos como uma "*coleção de computadores independentes que aparecem ao usuário do sistema como um único computador*". Por outro lado, Coulouris [14] descreve como "*coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados*".

As definições de sistemas distribuídos são bem próximas, mudando apenas a forma de se expressar de cada autor, com isso, pode-se encontrar algumas características básicas aos sistemas distribuídos que são: heterogeneidade, escalabilidade, transparência, concorrência e tolerância a falhas, os quais são descritos a seguir:

1. Heterogeneidade está relacionada ao fato dos sistemas distribuídos possuírem diferentes tipos de rede, hardware (diferentes representação de dados), protocolos de comunicação, etc, o que pode gerar vários problemas na integração. Então, para resolver o problema de heterogeneidade define-se uma camada, chamada *middleware*, que funciona como um software intermediador, utilizado para mover ou transportar informações e dados heterogêneos entre programas.

2. Escalabilidade é a capacidade que um sistema deve possuir de aumentar significativamente o número de usuários ou recursos disponíveis sem diminuir o desempenho.

3. Transparência está ligada ao fato de que os componentes do sistemas não podem ser vistos como partes independentes, o sistema em si deve ser percebido como um todo aos seus usuários.

4. Concorrência é a característica responsável por dar suporte e coordenar múltiplos acessos simultâneos a recursos compartilhados, mantendo a ordem.

5. Tolerância a falhas não remete que nenhuma falha ocorrerá, pois falhas são inevitáveis. A tolerância significa conter os efeitos que uma falha pode ocasionar de maneira que o sistema continue funcionando.

Clusters e Grades computacionais, além de antecederem ao conceito de Computação em Nuvem, são paradigmas de sistemas distribuídos importantes e tidos como base para o conceito de nuvem computacional. Esses paradigmas serão abordados nas próximas seções.

2.1.1 *Cluster* de Computadores

Buyya *et al.* [15] defende que devido a queda de preços e a popularização de computadores pessoais e/ou estações de trabalhos, e com o surgimento de ambientes multiprocessados de alto desempenho e avanços de redes de alta velocidade, surgiu um importante desenvolvimento tecnológico em sistemas distribuídos, que é a clusterização de computadores.

Pela definição de Morrison [16], os *clusters* computacionais são computadores relativamente simples que através de uma rede de alto desempenho podem parecer um único super computador, ou seja, pode se utilizar a programação paralela em que um único programa é executado em paralelo nas várias máquinas interligadas.

Com o surgimento da clusterização e a possibilidade de integrar uma coleção de computadores de maneira transparente, surge a ideia de entregar recursos computacionais para partes interessadas. Entretanto, para a construção de um *cluster*, os computadores integrados devem possuir elementos essenciais, tais como [16]:

1. Serem computadores de alta performance, preferencialmente servidores, *Mainframes* e/ou *Workstations*;

2. Possuir Sistema Operacional;

3. Conexão com uma rede de alta performance;

4. *Middleware* de gerenciamento para abstrair e facilitar o desenvolvimento e uso de aplicações;

5. Ambiente de computação paralela;

A Figura 2.1 apresenta a arquitetura conceitual de um *cluster* com os elementos indicados anteriormente. Nesta arquitetura o *cluster* é formado por um nó gerente que controla vários outros nós computacionais através de bibliotecas de paralelismo, funcionando como um *middleware* para executar os programas em paralelo.

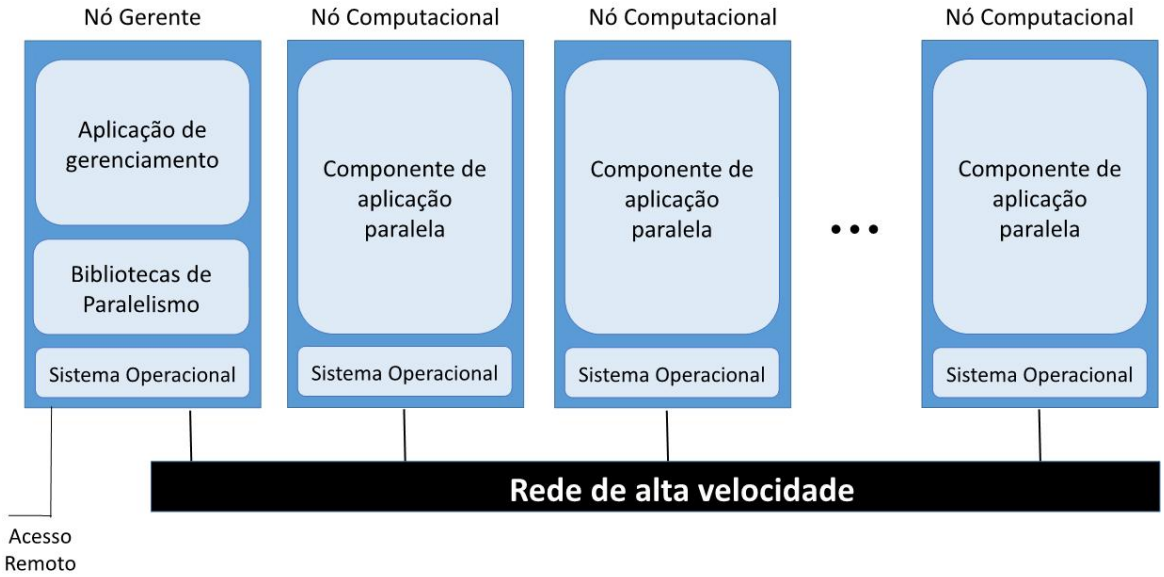

Figura 2.1: Exemplo de *Cluster* Computacional, adaptado de [1].

Segundo Tanenbaum [1], outra característica importante de *clusters* é a homogeneidade, ou seja, computadores interligados devem possuir a mesma arquitetura e características, utilizar o mesmo sistema operacional, estarem ligados a uma mesma rede, etc. Diferentemente, as grades de computadores são heterogêneas e serão abordadas na próxima seção.

2.1.2 Computação em Grade

O ponto chave deste sistema distribuído, segunto Foster *et al.* [17], é que os recursos de diferentes organizações (indivíduos e/ou organizações interessados) são reunidos para permitir a colaboração das partes formando uma organização virtual. Pessoas de uma mesma organização virtual tem acesso a recursos como servidores de computadores (pode

ter acesso a um *cluster* por exemplo), banco e armazenamento de dados e outros itens mais específicos como sensores, telescópios, etc., os quais são providos pela organização.

Apesar das grades de computacionais ser um paradigma que surgiu, inicialmente, com a mesma ideia dos *clusters* a de integrar uma coleção de computadores de maneira transparente, uma característica que diferencia as duas plataformas é que os *clusters* são homogêneas e as grades são heterogêneas [1].

A heterogeneidade permite que nenhuma prognose precise ser feita sobre características de hardware, sistema operacional, rede, etc. Isto é, as grades de computadores permitem acesso a recursos computacionais de diferentes fonte e domínios, por exemplo, diferentes organizações podem compartilhar recursos de modo colaborativo entre si.

Com isso, o foco do ambiente de grade computacional é o compartilhamento de grandes quantidades de recursos e que, geralmente, pode ser orientado à computação de alta performance [18]. Ou seja, é uma forma de computação que envolve coordenar e compartilhar recursos computacionais, aplicações, dados, armazenamento e/ou recursos de rede através de sistemas dinâmicos e geograficamente dispersos.

Neste contexto, as grades devem possuir uma arquitetura capaz de administrar e gerenciar os recursos de diversos sistema. Assim, deve ser capaz de lidar com sistemas operacionais, protocolos envolvidos, aplicações que rodam no sistema, utilização de memória, consumo de CPU e diversos outros fatores que envolvem a grade. Foster *et al* [2] dividem a arquitetura em cinco camadas, como mostra a Figura 2.2.

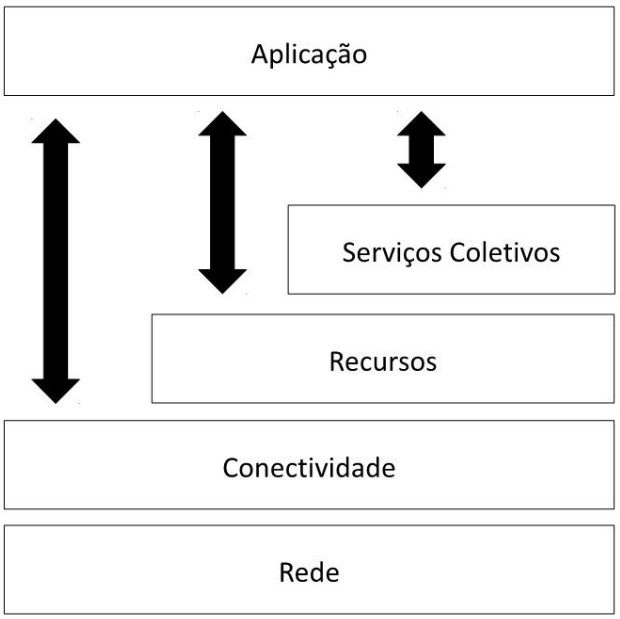

Figura 2.2: Arquitetura de Grade Computacional, adaptado de [2].

Na arquitetura apresentada, cada camada possui suas características e objetivos, as quais são:

1. Camada de Aplicação tem o objetivo de gerenciar aplicações que operam dentro de uma organização virtual, ou seja, gerenciar aplicações que compartilham os mesmos recursos;
2. Camada de Serviços Coletivos é responsável por capturar e gerenciar transações entre conjuntos de recursos;
3. Camada de Recursos define protocolos para transações, monitoramento e controle de operações que utilizam recursos individuais, ou seja, a camada apenas se importa com o seu grupo de recursos, e não com o estado global dos demais recursos compartilhados;
4. Camada de Conectividade é a camada onde a comunicação e os protocolos de autenticação para transações específicas da rede interna são definidos. Também deve prover mecanismos de segurança para verificar a identidade dos usuários e dos recursos;
5. Camada de Rede deve fornecer acesso compartilhado aos recursos (uma entidade lógica, um*cluster*, etc.) computacionais a partir dos protocolos da grade;

Assim sendo, o paradigma de grades computacionais é o que mais se parece com nuvens computacionais, pois se assemelham em tentar reduzir custo de computação, aumentar a flexibilidade e a entregar de recursos computacionais para agentes interessados. A Seção 2.2 abordará os conceitos relacionados à computação em nuvem.

2.2 Computação em Nuvem

Na literatura pode-se se encontrar diversas definições para o paradigma de computação em nuvem. O NIST (*National Institute of Standards and Technology*) [12] define computação em nuvem como um modelo para acesso conveniente, sob demanda, e de qualquer localização, a uma rede compartilhada de recursos de computação (isto é, redes, servidores, armazenamento, aplicativos e serviços) que possam ser prontamente disponibilizados e liberados com um esforço mínimo de gestão ou de interação com o provedor de serviços.

De acordo com Buyya *et al.* [19] nuvem é um sistema de computação paralela e distribuída que consiste em um conjunto de computadores interligados e virtualizados, que são dinamicamente providos e apresentados como um ou mais recursos de computação unificados, e é baseada em contratos de níveis de serviço estabelecidos através de negociação

entre o prestador de serviço e os consumidores. E de maneira mais genérica, Armbrust *et al.* [20] definem nuvem como um centro de dados de hardware e software que prove serviços.

Diante dos conceitos apresentados, pode-se observar semelhanças com outros sistemas distribuídos e a sobreposições dos conceitos. Foster *et al.* [2] representam a sobreposição conforme a Figura 2.3, de maneira que os conceitos de sistemas distribuídos de *cluster*, grade e nuvem computacional estão interligados.

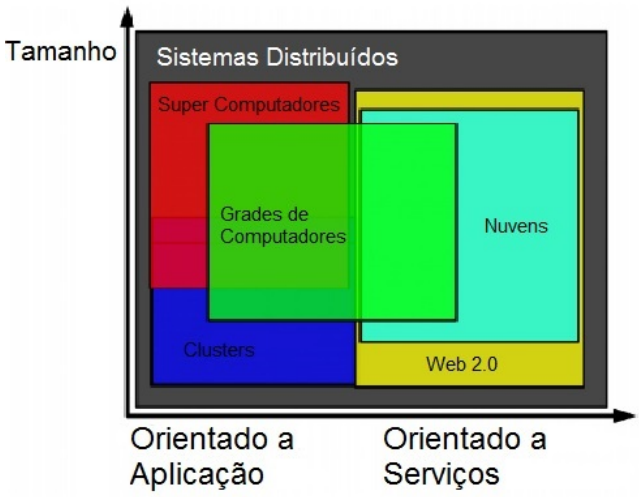

Figura 2.3: Nuvens Comparadas a outros Sistemas Distribuídos, adaptado de [2].

Entretanto, há um ponto que a computação em nuvem se distinguem dos demais paradigmas, no qual, a computação em nuvem se baseia em outro fundamento da computação que é a computação utilitária. Segundo Rappa [21] a computação utilitária pode ser definida como um modelo que "*entrega sobre demanda de infraestrutura, aplicações e processos de negócios de maneira segura, compartilhada e escalável em um ambiente de computação através da internet sobre uma taxa*".

Portanto, unindo todas as diferentes definições de computação em nuvem na literatura, e sabendo que computação utilitária e grades de computadores fundamentam este paradigma, algumas características levam a um ponto em comum, a de que computação em nuvem deve fornecer de maneira simples, rápida, dinâmica e sobre demanda serviços e recursos de computação para clientes interessados, através da internet.

Nesse contexto, algumas características são essências para caracterizar e entender o funcionamento de nuvens. Para isso, o NIST [12] listou cinco principais características: auto-atendimento sobre demanda, amplo acesso à rede, *pool* de recursos, elasticidade rápida e medição de serviço, que servem para dar suporte, em alto nível, para o entendimento de como as nuvens são projetadas. As cinco principais características são definidas como:

1. Auto atendimento sobre demanda (*On-demand self-service*):

Esta característica está ligada a capacidade de prover opções aos usuários, de acordo com suas vontades, de aumentar ou diminuir recursos computacionais, tais como: tempo de servidor e armazenamento de rede, automaticamente e unilateralmente. Ou seja, não há necessidade de interação com humanos prestadores de serviço para atender demandas de usuários.

2. Amplo acesso à rede (*Broad network acess*):

   Um aspecto importante para nuvem é a sua conectividade, ou seja, uma nuvem deve ser capaz de fornecer acesso para diferentes dispositivos (*smartphones*, *tablets*, notebooks, etc.) e diferentes sistemas operacionais. Então, uma nuvem deve garantir que todos os seus serviços possam ser acessados de maneira padronizada por diferentes dispositivos, para promover uma interação heterogênea para seus usuários.

3. *Pool* de recursos (*Resource pooling*):

   Exemplos de recursos que uma nuvem pode gerenciar são armazenamento, processamento, memória e largura de banda, podendo ocorrer variação de acordo com o objetivo para o qual a nuvem foi criada. Apesar dos recursos poderem variar, nuvens devem atender dinamicamente e conforme a demanda dos usuários à redistribuição dos recursos. Essa característica também leva em consideração a transparência, ou seja, usuários não podem ter o controle e nem o conhecimento de onde exatamente está a localização desses recursos, podendo apenas ter informações em um nível maior de abstração sobre a localização dos recursos, como: em qual país, estado, ou qual centro de dados se encontram.

4. Elasticidade rápida (*Rapid elasticity*):

   A elasticidade está diretamente associada as características 1 e 3, e pode ser a mais atrativa para serviços de nuvem. Quando se fala que os usuários podem redistribuir dinamicamente recursos computacionais sem que haja interação com humanos de maneira abstrata e de acordo com sua vontade, espera-se que essa interação seja a mais fluida possível. Então, essa característica promove a necessidade de aumentar ou diminuir recursos de maneira rápida, e dar a sensação de que os recursos são infinitos, ou seja, para os usuários os recursos devem ser sem limites e podem ser modificados a qualquer hora.

5. Medição de serviço (*Measured service*):

   Os serviços oferecidos podem ser monitorados, controlados e mensurados. Uma nuvem deve fornecer transparência tanto para os usuários monitorarem os gastos que possuem e controlar seus recursos, quanto para os provedores de serviços para otimizar o controle e verificar a demanda dos recursos oferecidos.

2.2.1 Arquitetura

Zhang *et al.* [3] defendem que a arquitetura de uma nuvem de computadores pode ser dividida em quatro camadas: camada de hardware (ou camada de *datacenter*), camada de infraestrutura, camada de plataforma e camada de aplicação. As quais são detalhadas a seguir:

1. Camada de hardware:

   Essa Camada é responsável pela gerencia dos recursos físicos da nuvem, como roteadores, servidores, *switches*, sistemas de energia e de refrigeração.

2. Camada de infraestrutura

   Camada essencial para atender as características básicas descritas pelo NIST [12]. Para isso, a camada de infraestrutura (também chamada de camada de virtualização) cria um *pool* de armazenamento e de recursos computacionais através da partição dos recursos físicos utilizando tecnologias de virtualização.

3. Camada de plataforma:

   A finalidade da camada de plataforma é minimizar a carga de implantação de aplicativos diretamente em *containers*, por exemplo, a camada pode fornecer *frameworks* para interfaces de aplicações e para a implementação de armazenamento, banco de dados e lógica de negócios de aplicações web.

4. Camada de software:

   A camada de software consiste em aplicações que são executadas em nuvem, isso significa que, diferentemente de aplicações tradicionais, essas fazem uso do conceito de elasticidade que a nuvem proporciona para alcançar melhor desempenho e custos operacionais favoráveis.

As camadas apresentadas são propostas como uma hierarquia, na qual a mais alta é a camada de software e a mais baixa é a camada de hardware, como mostra a Figura 2.4. Assim, a arquitetura é pensada de maneira em que cada camada possa ser implementada como serviço à camada acima.

2.2.2 Modelos de Serviço

Cada nuvem é projetada para fornecer certos serviços, os três principais são: infraestrutura como serviço ou IaaS (*Infrastructure as a Service*), plataforma como serviço ou PaaS (*Platform as a Service*) e software como serviço ou SaaS (*Software as a Service*).

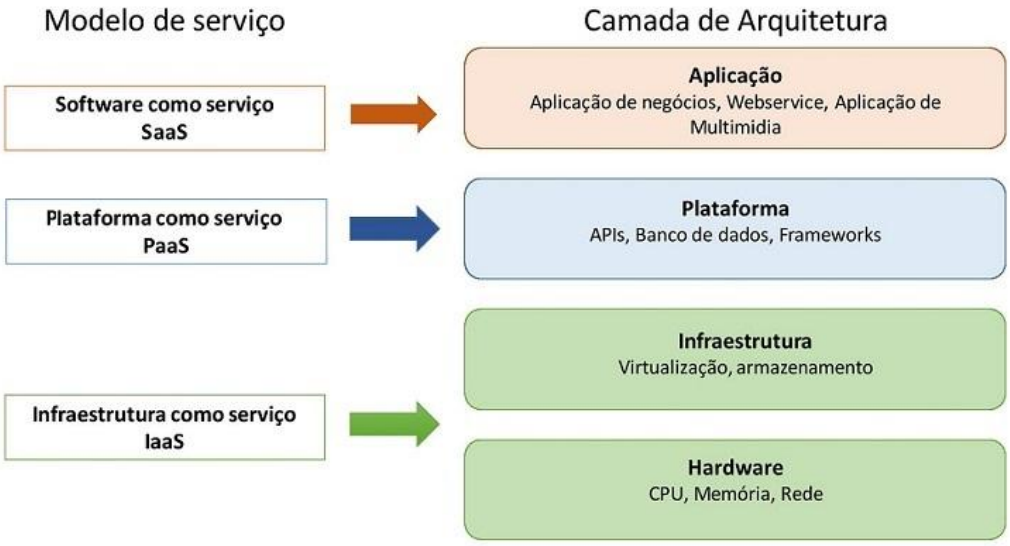

Figura 2.4: Arquitetura e Modelo de Serviço, adaptado de [3].

A Figura 2.4 mostra a relação entre os modelos de serviço e a arquitetura hierárquica apresentada na seção anterior.

O modelo de infraestrutura como serviço é considerado o modelo que fornece os serviços mais abaixo na camada de arquitetura. Foster *et al.* [22] definem que o modelo deve oferece recursos computacionais sobre demanda, como, por exemplo, poder computacional, armazenamento e rede.

Neste modelo o usuário tem controle sobre sistemas operacionais, aplicações e recursos computacionais. Porém, não devesse preocupar com o gerenciamento da infraestrutura, ou seja, o gerenciamento dos servidores, do armazenamento e/ou como os recursos são compartilhados.

O conceito de virtualização é essencial para provedores de IaaS, por isso, será apresentado de forma mais ampla no Capítulo 3. De uma maneira em geral, a virtualização permite que fornecedores de serviço de infraestrutura possam compartilhar recursos, entre diversos usuários, de uma mesma máquina física. Atualmente, grandes empresas fornecem nuvens de IaaS, por exemplo: Amazon EC2[1], Google *Compute Engine*[2] e GoGrid Datapipe[3].

No modelo de plataforma como serviço a camada de infraestrutura fica oculta aos usuários, isso significa que não devem se preocupar com o controle de recursos de infraestrutura como, por exemplo, rede, armazenamento, sistema operacional, memória,

[1] `https://aws.amazon.com/pt/ec2/`
[2] `https://cloud.google.com/appengine/`
[3] `https://www.datapipe.com/gogrid/`

etc. Ou seja, os serviços ofertados neste modelo devem prover a capacidade de implantar aplicações de usuários sem ter que gerenciar aspectos de hardware e de software.

Coyne *et al.* [23] citam alguns exemplos de serviços que são fornecidos a partir do modelo PaaS: *middleware*, servidores de banco de dados, servidores de aplicação e ambientes de desenvolvimento de softwares. Algumas nuvens que disponibilizam esse modelo são: Google *App Engine*[4], HEROKU[5] e Amazon RDS[6].

Por último, o modelo de software como serviço prove aplicações prontas para o uso, ou seja, o usuário deve apenas conectar com a aplicação que está executanto remotamente. Assim, SaaS elimina a necessidade de instalar e executar aplicações direto nos computadores dos usuários.

Os usuários de SaaS buscam um serviço de computação final e não propor ou desenvolver soluções computacionais. Por isso, os provedores de serviços devem gerenciar a infraestrutura, a plataforma que a aplicação está em execução e a própria aplicação. Alguns exemplos de nuvens que oferecem SaaS são: ownCloud[7], CRM SalesForce[8] Microsoft Office *online*[9].

2.2.3 Papéis na Computação em Nuvem

Os papéis são importantes para entender como funciona a dinâmica entre os diferentes tipos de serviços apresentados na Seção 2.2.2. De acordo com Marinos e Briscoe [4], exitem três papéis principais, e a partir deles pode-se definir responsabilidades, tipos de acessos e perfis para os diferentes usuários que estão envolvidos em uma solução de computação em nuvem. Esses papéis são provedor, desenvolvedor, e usuário final.

Os provedores são responsáveis por toda a gerência das soluções nuvem. Eles podem fornecer qualquer um dos tipos de modelos de serviços (IaaS, PaaS e/ou SaaS), e são encarregados de toda a infraestrutura e as características que a nuvem precisas prover.

Os desenvolvedores possuem um papel misto, pois podem tanto fornecer quanto consumir serviços. Os desenvolvedores fornecem serviços de software (SaaS) para usuários finais, mas também podem atuar como usuários finais quando consomem serviços de plataforma e de infraestrutura (PaaS e IaaS).

Por último, os usuários finais, como o próprio nome sugere, possuem o papel de consumir serviços. Normalmente, os usuários que não possuem familiaridade com computação e desenvolvimento consomem SaaS, e os usuários finais mais experientes e familiarizados

[4] `https://cloud.google.com/appengine/`
[5] `https://www.heroku.com/`
[6] `https://aws.amazon.com/pt/rds/`
[7] `https://owncloud.org/features/`
[8] `http://www.salesforce.com/br/what-is-salesforce/`
[9] `https://www.office.com/`

consomem PaaS e/ou IaaS. A Figura 2.5 representa a interação que ocorre com os papéis principais (provedor, desenvolvedores e usuário final), e os diferentes tipos de serviços (Iaas, PaaS e SaaS).

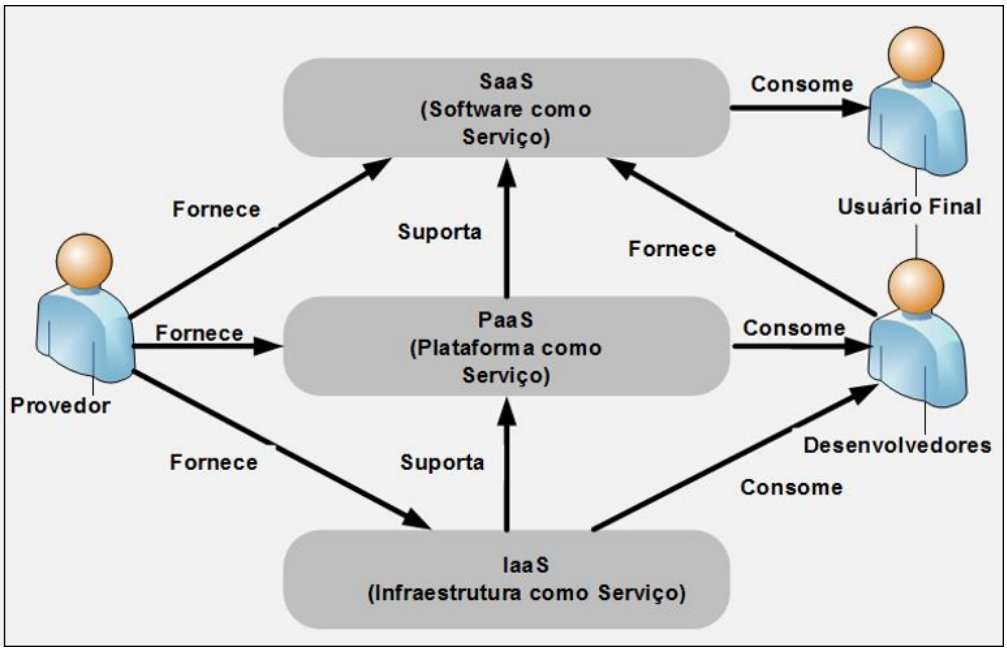

Figura 2.5: Papéis na Computação em Nuvem, adaptado de [4].

2.2.4 Modelos de Implantação

Independente do serviço oferecido, existem diferentes maneiras de se implantar uma nuvem computacional. A escolha a ser adotada dependerá da necessidade dos usuários e/ou de como o provedor pensa em fornecer o seu serviço. O NIST [12] define quatro diferentes modelos de implantação: público, privado, comunitário e híbrido.

O modelo público de implantação visa serviços de propósito mais geral e, geralmente, seu uso é gerido pelo método pague o quanto usar (*pay-per-use*). Isso torna o modelo atrativo tanto para provedores com interesse de vender serviços, como para usuários que não precisam investir em recursos que podem ficar ociosos com o tempo, gastando somente quando necessário.

Este modelo, normalmente, é utilizado por empresas, *startups* ou setores de TI que fornecem serviços para empresas de outros ramos ou clientes interessados.

Por outro lado, uma nuvem privada é pensada quando se deseja que o uso seja exclusivo de uma organização ou um grupo de organizações, ou seja, a organização possui controle de como os recursos estão sendo utilizados, e de qual maneira estão sendo disponíveis.

Uma característica marcante é o controle que a organização que detém o uso da nuvem pode assumir diante dos mais variados recursos da nuvem, e de como ela pode ser

estruturada. O modelo de nuvem privada é construído, operado e mantido pela operadora da nuvem e compartilhados por toda a organização, sendo ideal para organizações que querem entrar no mundo da computação em nuvem mas não abrem mão do controle da mesma.

Outra característica importante é a segurança fornecida pela restrição de acesso. Taurion [24] defende que nuvens privadas possuem mecanismos de segurança e confiabilidade mais severos que nuvens públicas, pois são implantadas dentro do *firewall* das organizações.

Sendo assim, o modelo privado é ideal para organizações que não desejam concorrer com recursos de uma maneira pública e/ou desejam melhor segurança e/ou controle sobre os recursos computacionais oferecidos pelas nuvens.

O modelo comunitário de implantação é uma variação do modelo de nuvem privada, onde uma comunidade de organizações tem um objetivo em comum. O compartilhamento de recursos (espaço de armazenamento, poder computacional, softwares, etc.) da nuvem pode gerar concorrência e disputas entre organizações interessadas. Entretanto, este modelo leva a diminuição dos custos na implantação dos serviços da nuvem pelo fato de compartilharem recursos.

Nuvens comunitárias são indicadas para organizações com interesses em comum, por exemplo, organizações com o mesmo requisito em segurança e/ou em performance podem se beneficiar com este modelo de implantação.

Por último, o modelo híbrido, como o próprio nome sugere, é a junção de dois ou mais modelos de implantação. Essa é uma solução útil quando deseja-se unir os benefícios que cada modelo pode proporcionar, por exemplo, quando se deseja trafegar dados sensíveis e sigilosos em um ambiente controlado e gerenciado internamente (utilização de uma nuvem privada), enquanto outros dados e aplicações podem trafegar em nuvens públicas ou comunitárias.

2.3 Considerações Finais

Neste capítulo foram apresentados os principais conceitos relacionados a computação distribuída. Primeiramente, foram abordados os paradigmas de *cluster* e grade computacional. Em seguida, o foco foi apresentar uma visão sobre os principais conceitos, características, definições e termos da computação em nuvem.

O Capítulo 3 abordará, inicialmente, o conceito de virtualização de hardware, o qual é fundamental para o modelo de serviço IaaS. Em seguida, será apresentado um estudo sobre as principais plataformas de implantação de nuvens por meio do modelo de serviço IaaS.

Capítulo 3

Virtualização e Plataformas para Nuvens de IaaS

Nuvens de IaaS devem fornecer serviços como espaço de armazenamento, recursos computacionais e de rede. Então, para projetar uma nuvem de IaaS deve-se considerar vários fatores, além de atender as características essênciais citadas no Capítulo 2.

A forma de contratação de serviços virtuais, entre usuários e provedores, é baseada no uso. Neste modelo o fornecimento de serviço aos usuários podem ser cobrado de acordo com o consumo. A medição de uso dos recursos pode ser por tempo requerido, por recursos utilizados, pela quantidade de dados trafegados e/ou dados armazenados, ou uma combinação dos recursos oferecidos. Por isso, é necessário que haja o gerenciamento da nuvem de e dos recursos disponibilizados por cada usuário.

Assim sendo, como dito, na Seção 2.2.2, nuvens de IaaS utilizam a virtualização de hardware para permitir o compartilhamento de recursos de uma única máquina física entre diversos usuários.

Nesse contexto, este capítulo tem o objetivo de conceituar a virtualização de hardware utilizada por nuvens de IaaS, como também, apresentar as principais plataformas e soluções para a implantação e o gerenciamento de nuvem de infraestrutura.

3.1 Virtualização de Hardware

Nuvens de IaaS são, geralmente, apoiadas por centros de dados e servidores, e podem ser compostas por apenas algumas unidades, até milhares de computadores. Tais servidores são construídos para servir muitos usuários. Nesse contexto, a virtualização de hardware deve ser considerada fundamental para que os múltiplos usuários possam acessar recursos de uma mesma máquina real.

A ideia de virtualização já é bastante definida na literatura. Buyya *et al.* [5] definem uma máquina virtual (*Virtual Machine*) como sendo um sistema operacional, ou um ambiente de aplicação que está instalado em software que imita um hardware dedicado. Isto é, o usuário final tem a mesma experiência em uma máquina virtual como teria em uma máquina dedicada, e permite que organizações possam trabalhar com diversas plataformas de software, sem a necessidade de aumentar o número de máquinas físicas. Outra característica deste tipo de tecnologia é o compartilhamento dos recursos, como, por exemplo, processadores, memória RAM, interfaces de rede e discos, da máquina física com todas as máquinas virtuais ali presentes.

Com isso, surgiu uma nova tecnologia para gerenciar e monitorar máquinas virtuais (VM), os hipervisores. Os hipervisores, segundo Neiger *et al.* [25], são um intermediador entre o hardware e as VMs, e controlam o acesso ao hardware físico para cada sistema operacional convidado (para cada VM convidada). A Figura 3.1 ilustra o funcionamento da virtualização de hardware, um hipervisor gerencia várias VMs em um único hardware.

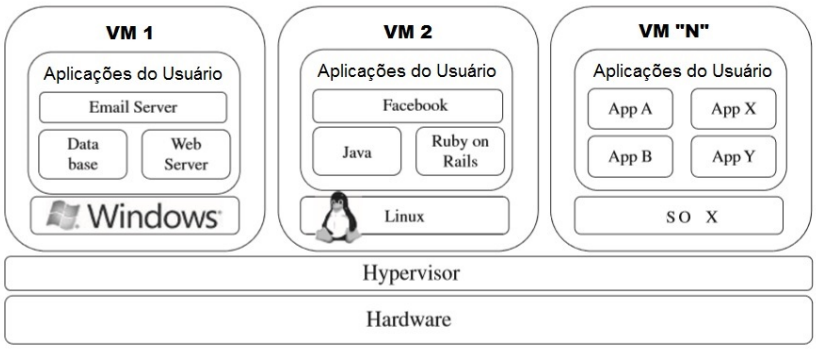

Figura 3.1: Exemplo de Virtualização de Máquinas, adaptado de [5].

Buyya *et al.* referencia três notáveis hipervisores, os quais são:

1. VMWare ESXi[1] é um dos pioneiros no mercado de virtualização. Ele é instalado diretamente no servidor físico, enquanto outros podem exigir um sistema operacional hospedeiro. Ele fornece técnicas avançadas de virtualização de processador, memória e E/S.

2. Xen[2] serviu de base para vários outros hipervisores comerciais e de código aberto. Este hipervisor foi um dos pioneiros na técnica de para-virtualização, que implica na utilização de *kernel* especializado para a interação com hipervisores aumentando significativamente suas performance.

[1] `http://www.vmware.com/`
[2] `http://www.xenproject.org/`

3. KVM[3] é um hipervisor baseado em no *kernel* Linux e é suportado nativamente em diversas distribuições. O KVM utiliza virtualização assistida por hardware, que melhora o desempenho e permite que o suporte de diversos sistemas operacionais como Windows, Linux e UNIX.

3.2 Plataformas de Impantação de IaaS

Atualmente, existem diversas plataformas para a implantação de nuvens de IaaS. Griebler *et al.* [26] citam algumas das mais conhecidas (comerciais e de código aberto): Abiquo [27], Apache Virtual Computing Lab [28], Apache CloudStack [29], ConVirt Enterprise Cloud [30], HPE Helion Eucalyptus [10], Nimbus Infrastructure [31], OpenNebula [9], OpenQRM [32], OpenStack [33] e Ubuntu Cloud [34]. Os quais são descritas a seguir:

1. Abiquo:

   Abiquo [27] visa criar nuvens privadas baseadas em uma infraestrutura já existente ou controlar o uso de serviços em nuvem pública. É uma ferramenta comercial, constituída pelo gerenciador de rede, *cluster*, servidor, rede de armazenamento, serviços remotos e um servidor de armazenamento. Essa solução fornece *logs* para analisar os recursos, e possui um mecanismo de preços que atribui um valor a qualquer recurso (CPU, RAM, armazenamento, etc.).

2. Apache Virtual Computing Lab (VCL):

   Apache VCL [28] é uma plataforma de código aberto para a computação em nuvem, com o objetivo de entregar um ambiente de computação personalizado e dedicado. Essa plataforma oferece ambiente de máquina virtual ou até mesmo um *cluster* de servidores físicos. É utilizado para acesso remoto a partir da Internet para reservas de recursos computacionais. Ele tem sua arquitetura formada por portal web, banco de dados, nós de gestão e nós de computação.

3. Apache CloudStack:

   O CloudStack [29] é uma plataforma de código aberto e foi desenvolvida para implantar e gerenciar grandes redes de máquinas virtuais, pois possui escalabilidade e alta disponibilidade de infraestrutura. Ela oferece a criação de nuvens públicas, privadas ou híbridas com serviço de infraestrutura para os usuários através de uma interface Web.

4. ConVirt Enterprise Cloud:

[3] `http://www.linux-kvm.org/page/Main_Page`

O ConVirt [30] é uma plataforma comercial que fornece serviços de infraestrutura permitindo a virtualização, o gerenciamento de nuvem privada e a integração com nuvens públicas. Os serviços de infraestruturas fornecidos podem ser entregues e provisionados sobre demanda via Internet a clientes interessados.

5. HPE Helion Eucalyptus:

   O Eucalyptus [10] é uma solução de código aberto para criar nuvens privadas e híbridas. A ferramenta é indicada para computação em nuvem em ambientes de computação empresarial corporativa, pois permite uma fácil integração de nuvens públicas com nuvens privadas, oferecendo um ambiente de nuvem híbrida.

6. Nimbus Infrastructure:

   Nimbus [31] é um projeto de código aberto que permite clientes usarem recursos de computadores remotamente, provendo máquinas virtuais como solução de infraestrutura como serviço. Também possui um serviço de armazenamento que pode ser usado separadamente ou integrado a si.

7. OpenNebula:

   O projeto de código aberto OpenNebula [9] prove soluções flexíveis para o gerenciamento de centros de dados virtualizados, e possibilita o gerenciamento de nuvens privadas, públicas e híbridas de infraestrutura como serviço. Foi desenvolvido para uma gestão mais eficiente e escalável de máquinas virtuais em infraestruturas distribuídas.

8. OpenQRM

   OpenQRM [32] é um gerenciador de nuvens de computação de código aberto que gerencia virtualização, armazenamento, a rede e toda a infraestrutura de TI a partir de um console. Ele permite a criação de nuvens privadas, públicas e híbridas com alta disponibilidade.

9. OpenStack

   OpenStack [33] é uma plataforma de código aberto de nuvem que controla uma grande quantidade de recursos computacionais, armazenamento e recursos de rede através de um gerenciador que utiliza interfaces web, permitindo criar nuvens privadas, públicas e híbridas de infraestrutura.

10. Ubuntu Cloud

    A plataforma Ubuntu Cloud [34], chamada anteriormente de Ubuntu Enterprise Cloud, foi baseado inicialmente na plataforma Eucalyptus. Atualmente, o Ubuntu

Cloud é baseado em um conjunto de ferramentas para facilitar a gestão do ambiente de nuvem, sendo que a plataforma CloudStack é a pricipal delas.

Apesar de todas as plataformas servirem ao mesmo propósito, o de implantação de nuvem de IaaS, elas possuem diferentes estratégias. A fim de definir critérios que pudessem influenciar na escolha da plataforma a ser adotada na implantação da nuvem privada neste projeto, a Seção 3.2.1 apresenta algumas características a serem analisadas nas principais plataformas encontradas na literatura.

3.2.1 Características Analisadas

A relevância e a qualificação das características de plataformas de IaaS podem variar. Assim, diferentes estratégias devem ser analisadas ao projetar uma nuvem de IaaS. Para que essa análise seja feita, Laszewski *et al.* [35] adotaram uma metodologia para qualificar nove características principais, que podem ser usadas para comparar as diversas plataformas de IaaS. Por outro lado, Morrison *et al.* [36] utilizaram vinte características para comparação das soluções de IaaS.

Para analisar a plataforma que melhor se adéqua a este trabalho foi utilizada uma mescla dos dois estudos, já que algumas características selecionadas são similares, e/ou têm pouca relevância para a qualificação de plataformas.

Nesse contexto, foram escolhidas dez características a serem avaliadas: interface de usuário, suporte de API, hipervisor, arquitetura, rede, armazenamento e elasticidade, autentificação, lançamento de versões, licença e documentação. Essas características são melhor explicadas a seguir:

1. Interface de usuário:

   A interface proporciona a interação dos usuários com a ferramenta, os diferentes softwares podem trazer diversas abordagens para proporcionar essa interação. As interfaces tipicamente usadas são *Command Line Interface* (CLI) ou interfaces *Browser User Interface* (BUI).

2. Suporte de API:

   API (*Application Programming Interface*) de nuvem são interfaces de programação de aplicação que são usadas para construir aplicações. Segundo Wen *et al.* [37], as APIs de infraestrutura como serviço ajudam a controlar e destribuir recursos específicos, assim como, auxiliar a configuração da nuvem.

   Quando se fala que uma nuvem dá suporte a alguma API, significa que a nuvem utiliza os serviços e/ou permite o acesso as funcionalidades de outras APIs, ou seja, suporta padrões e rotinas já estabelecidas por outra interface.

Nos estudos realizados por Laszewski *et al.* [35] e Morrison *et al.* [36], o suporte as APIs da Amazon Web Service *AWS*[4] mostrou-se dominante entre as plataformas de IaaS. As APIs mais encontradas nesses estudos foram: Amazon Elastic Compute Cloud (Amazon EC2)[5] e o Amazon Simple Storage Service (Amazon S3)[6].

O Amazon EC2 fornece capacidade de computação redimensionável na nuvem, ou seja, oferece um controle dos recursos computacionais, permitindo o escalonamento rápido de recursos de acordo com a necessidade. Já Amazon S3, é uma API que oferece armazenamento de objetos através de interface web para armazenar e recuperar dados de qualquer parte da Internet.

3. Hipervisor:

   Como visto na Seção 3.1, um hipervisor é uma plataforma que permite aplicar diversas técnicas de controle de virtualização para utilizar diferentes sistemas operacionais. As plataformas de nuvem como serviço oferecem suporte a diferentes tipos de hipervisores para proporcionar uma maior flexibilidade ao projeto da nuvem.

   Entretanto, as plataformas de nuvem de IaaS fornecem diferentes recursos para cada hipervisor. Os hipervisores mais frequentemente suportados são [35] [36]: KVM[7], Xen[8], VMware[9] e Hyper-V[10].

4. Arquitetura:

   Os softwares podem ser estruturados de diferentes maneiras, por isso é importante saber como é a arquitetura, se está dividido em módulos ou os serviços dispostos são oferecidos por um núcleo centrado. A arquitetura das plataformas mostram os detalhes de como ela foi construída e de como ela opera.

5. Rede:

   A rede pode ser gerenciada de várias maneiras em cada plataforma de IaaS, então, é essencial avaliar quais são elas e o que cada ferramenta pode oferecer. Pode-se considerar que a rede influência como os recursos vão ser utilizados e o gerenciamento do ambiente da nuvem.

   Nos estudos realizados por Laszewski *et al.* [35] e Morrison *et al.* [36], os principais conceitos de redes e gerenciamento de redes encontrados nas plataformas foram:

[4] `https://aws.amazon.com/pt/`
[5] `https://aws.amazon.com/pt/ec2/`
[6] `https://aws.amazon.com/pt/s3/`
[7] `http://www.linux-kvm.org/page/Main_Page`
[8] `http://www.xenproject.org/`
[9] `http://www.vmware.com/`
[10] `https://msdn.microsoft.com/pt-br/library/hh831531(v=ws.11).aspx`

VLAN (rede local virtual) [38], *Flat Networking* (alternativa mais simples de rede), DHCP (protocolo de configuração dinâmica de host) [39], DNS (sistema de gerenciamento de nome) [40], VPN (rede virtual privada) [41] e SDN (redes definidas por software) [42].

6. Armazenamento e Elasticidade:

   A virtualização de armazenamento, de acordo com Buyya *et al.* [5], significa abstrair armazenamento lógico de armazenamento físico. Ao consolidar todos os dispositivos de armazenamento disponíveis em um centro de dados, permite, por exemplo, a criação de discos virtuais independentes de dispositivo e da localização. Nos estudos realizados por Laszewski *et al.* [35] e Morrison *et al.* [36] as tecnologias mais comuns encontradas para o armazenamento da nuvem foram: o protocolo iSCSI (*Internet Small Computer Systems Interface*) [43], o sistema de arquivos NFS (*Network File System*) [44] e suporte a *Fibre Channel* (infraestrutura apropriada para comunicação de alta velocidade).

   O armazenamento é muito importante para uma nuvem de IaaS, pois é essencial saber como a ferramenta gerencia e maneja as informações e os dados sobre os recursos de uma nuvem computacional. Com isso, o armazenamento é crucial para a elasticidade, uma das características essências que definem a computação em nuvem, fazendo com que os usuários possam redistribuir dinamicamente recursos computacionais. A elasticidade dos recursos pode ser tanto horizontal, isto é, há a possibilidade de aumentar ou diminuir o número de instâncias (VMs), assim como é possível sua migração para novos nós de processamento quanto vertical, na qual há o redimensionamento de atributos computacionais de CPU, disco, rede ou memória, além da alternativa de alocar ou desalocar nós de computação.

7. Autenticação:

   Autenticação é o processo de verificação de usuários, ou seja, verificar se você é realmente quem você está dizendo quem é. A segurança para quem está utilizando e para quem fornece o serviço de nuvem é muito importante, por isso, essa característica visa analisar qual é o processo de autenticação, os protocolos e metodologias usadas para tal tarefa.

   Nos estudos realizados por Laszewski *et al.* [35] e Morrison *et al.* [36], o protocolo mais comum entre as plataformas foi o LDAP (*Lightweight Directory Access Protocol*) [45]. Outras alternativas encontradas para compor e/ou subistituir o LDAP foram o padrão X509 [46] para chaves públicas, a linguagem SAML [47] (*Security Assertion Markup Language*), assim como, APIs proprietárias exclusivas de cada plataforma.

8. Lançamento de versões:

   O lançamento de versões deve ser analisado para observar a frequência com que o software é atualizado. Além disso, também auxilia a percepção de como está o desenvolvimento da ferramenta e se sua manutenção é periódica.

9. Licença:

   Essa característica analisa se o projeto faz parte de alguma organização ou quem possui licença sobre o software. Refere-se as restrições do usuário na distribuição, modificação ou uso do software, com ou sem o pagamento de *royalties*.

10. Documentação:

    A maneira a qual a documentação está disponível, influência na experiência do usuário em utilizar uma plataforma, se é de fácil entendimento ou não. É importante manter a documentação alinhada com o lançamento de versões, pois auxilia a manutenção e o entendimento das novas atualizações.

No contexto deste trabalho, o qual envolve o ambiente universitário e de pesquisa, a característica básica para a escolha da ferramenta de implantação é que ela deve ser de código aberto para maior liberdade do projeto. Zhang *et al.* [48] defendem que soluções de código aberto são tipicamente usadas para implantar nuvens privadas e híbridas, pois podem ser modificadas pelos utilizadores para criar um único pacote funcional adequado para a oferta de infraestrutura como serviço proposta. Assim sendo, a próxima seção apresentará as principais plataformas de código aberto.

3.3 Principais Plataformas IaaS de Código Aberto

Uma pesquisa conduzida por Linux.com [6] e The New Stack [49], anunciada no dia 20 de agosto de 2014 no evento CloudOpen [50] em Chicago, feita com mais 550 pessoas, mostra as principais plataformas de nuvens de código aberto. Entre os projetos de código aberto os quatro melhores classificados foram: OpenStack, CloudStack, Open Nebula e Eucalyptus, como pode ser visto na Figura 3.2.

A mais votada foi a plataforma OpenStack [7], seguida por CloudStack [8], OpenNebula [51] e Eucalyptus [52]. A soma dos votos de outras plataformas ficou com aproximadamente dois por cento dos votos. Então, para a análise deste trabalho foram escolhidas as quatro primeiras colocadas.

Nas próximas subseções as plataformas serão apresentadas conforme as características escolhidas neste trabalho. Por fim, será feita uma avaliação, na subseção 3.3.5, para a escolha da plataforma a ser utilizada neste projeto.

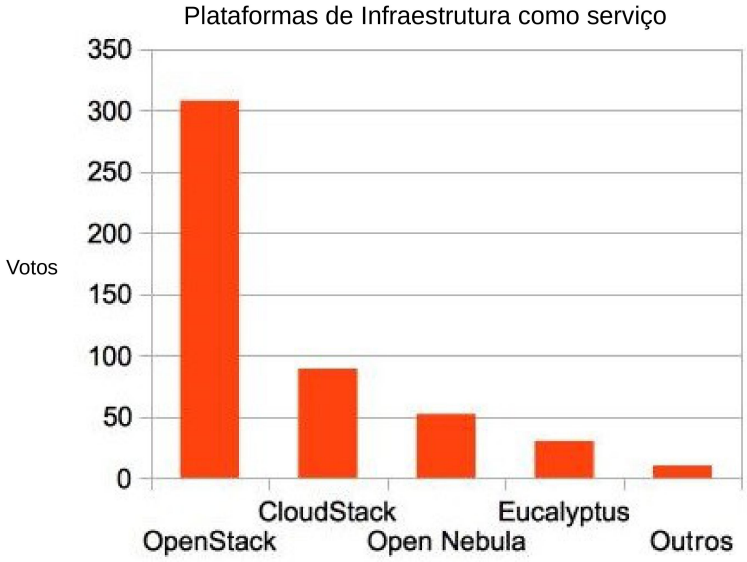

Figura 3.2: Pesquisa das Principais Plataformas para IaaS de Código Aberto, adaptado de [6].

3.3.1 OpenStack

O OpenStack é uma plataforma de código aberto com a licença Apache 2.0 para projetos de nuvens com o seu desenvolvimento realizado por uma comunidade de desenvolvedores. O ciclo de lançamento de melhorias e atualizações é em torno de 6 meses, durante esse tempo há a fase de planejamento que reuni a equipe de desenvolvedores para a programação das próximas atualizações.

A arquitetura da plataforma é dividida em subprojetos, cada projeto pode ser pensando como um módulo que fornece algum tipo de serviço, e também pode utilizar serviços de APIs como a Amazon EC2 e a Cloud Files REST[11]. A arquitetura conceitual conta com todos os projetos e a interação entre os projetos pode ser visto na Figura 3.3. Entretanto, outras estratégias podem ser utilizadas, como por exemplo, utilizar APIs para a substituição e/ou complementar a arquitetura da plataforma. Ao todo são 10 subprojetos, os quais são:

1. Horizon: provê uma interface web para a interação dos serviços do OpenStack, e oferece o serviço de painel de controle para as configurações da nuvem.

[11] `https://developer.rackspace.com/docs/cloud-files/v1/developer-guide/`

2. Nova: é o responsável pelo serviço computacional, com a função de gerenciar o ciclo de vida das instâncias computacionais, ajustando as máquinas virtuais conforme a demanda.

3. Neutron: é o subprojeto responsável pelo gerenciamento da rede e fornece serviços para os demais subprojetos que necessitam de gerenciamento de redes.

4. Swift: é o responsável pelo armazenamento e a recuperação de objetos de dados, sua tolerância a falhas ocorre pela replicação de dados.

5. Cinder: fornece armazenamento em bloco persistente para instâncias em execução. Sua arquitetura de blocos é conectável e facilita a criação e a gestão de dispositivos de armazenamento.

6. Keystone: provê serviços como autenticação e autorização, identificando as permissões e quem está utilizando os demais serviços.

7. Glance: cuida do armazenamento e da recuperação de imagens de disco das máquinas virtuais.

8. Ceilometer: monitora, realiza medições e afere a utilização dos recursos computacionais.

9. Heat: é o responsável por orquestrar integrações com diferentes APIs, pode ser considerado um serviço de alto nível.

10. Trove: não faz parte da arquitetura principal do OpenStack, mas é um subprojeto, de alto nível, que provê escalabilidade para o fornecimento de serviços de banco de dados.

A interface de usuário disponibilizada pela plataforma é intuitiva e pode ser tanto por comando CLI como por interface web BUI. Além disso, sua documentação é bem estrutura, mostra a visão geral da plataforma, seu funcionamento e sua integração.

O OpenStack oferece suporte para até oito hipervisores, os quais são: KVM, Xen, VMware, Hyper-V, Ironic[12], LXC[13], QEMU[14], Virtuozzo[15], vCenter[16] e XenServer[17]. Entretanto, nem todas as funcionalidades dos hipervisores estão disponíveis. Alguns possuem melhor integração com a plataforma, e ,por isso, deve se escolher aquele que se melhor se adapta ao projeto.

[12] http://docs.openstack.org/developer/ironic/
[13] https://linuxcontainers.org/lxd/
[14] http://wiki.qemu.org/Main_Page
[15] https://virtuozzo.com/
[16] https://www.vmware.com/support/mhm/doc/vcenter-multi-hypervisor-manager-10-release-notes.html
[17] https://www.citrix.com.br/products/xenserver/

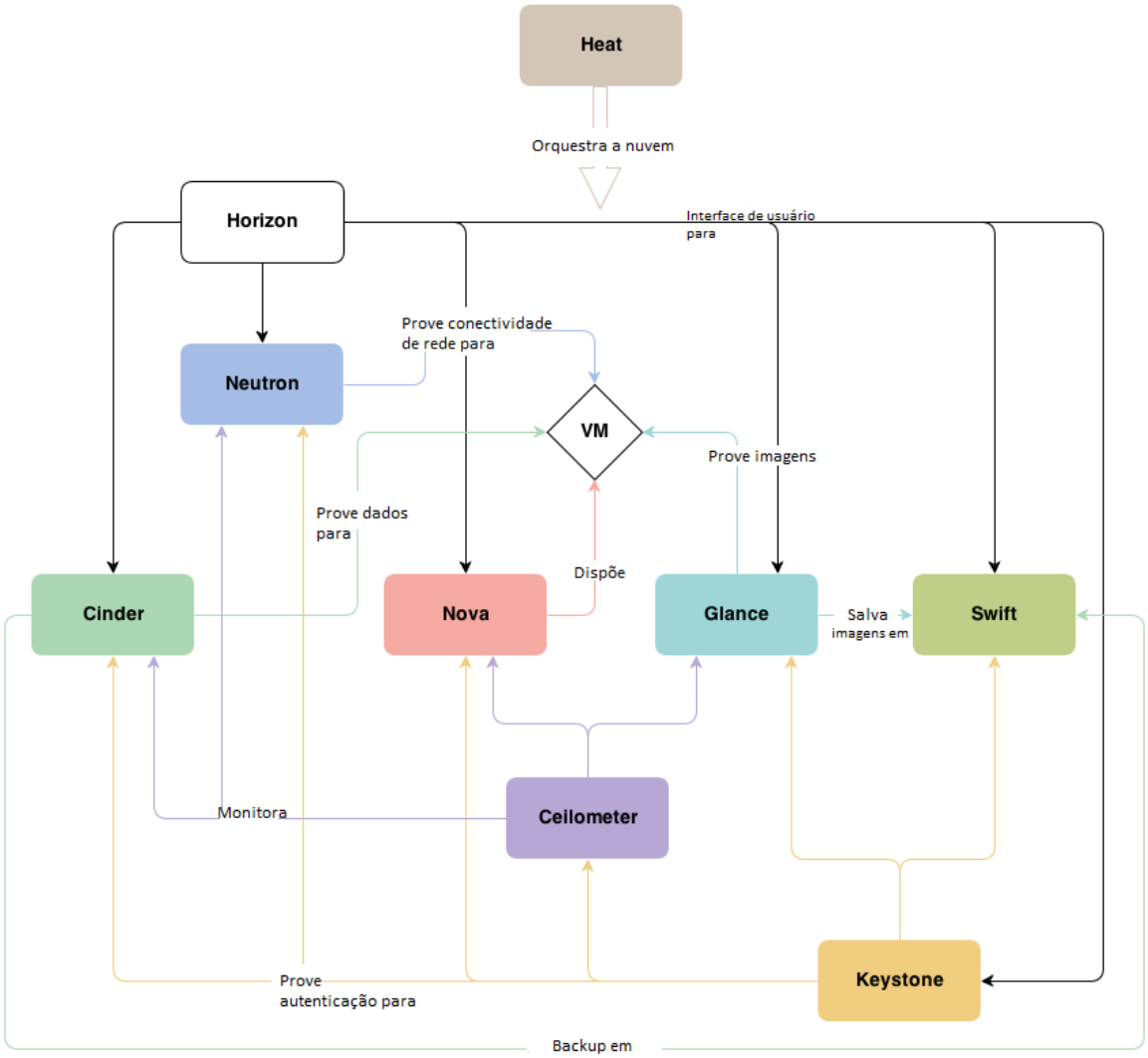

Figura 3.3: Fluxograma com a Interação da Arquitetura Conceitual do OpenStack, adaptado de [7].

A plataforma utiliza serviços de redes, tais como VLAN para distribuição de redes ou, alternativamente, *Flat Networking* (distribuição simplificada de redes). Também, há o suporte para protocolo DHCP, o qual é utilizado para prover endereços IP automaticamente, e também apresenta suporte para SDN.

Por outro lado, a autenticação, é fornecida como um serviço do módulo da arquitetura Keystone. Este módulo provê uma API de identificação que conta com os protocolos LDAP, Kerberos [53] e X509 [46].

Para o armazenamento o OpenStack aparece como um dispositivo de bloco que utiliza o protocolo iSCSI, a tecnologia *Fibre Channel*, o sistema de arquivos NFS ou uma série de outros protocolos proprietários. A elasticidade possui um escalonamento com valores padrões, ou seja, exitem *labels* predefinidos, no qual as máquinas virtuais possam crescer horizontalmente.

3.3.2 CloudStack

O CloudStack possui a licença Apache 2.0 e é uma plataforma de código aberto. O ciclo de atualização de versões é de quatro meses, no quinto mês a atualização passa por testes e é finalizada sua documentação, então, no sexto mês a versão é estabilizada. Com isso a documentação é frequentemente atualizada e possui todo o passo-a-passo para entender as principais características e o funcionamento da plataforma.

As principais APIs que a plataforma suporta são a Amazon EC2 e a Amazon S3. A sua arquitetura é hierárquica com quatro componentes principais, os quais são gerenciador, zonas, *pods* e *clusters*, detalhados a seguir:

1. Gerenciador: é a parte responsável pelo gerenciamento da nuvem e todas as camadas da arquitetura.

2. Zona: as zonas são onde os banco de dados e toda a organização lógica estão localizados, e tem o objetivo de prover redundância e isolamento. Cada zona permite ter sua política de gerenciamento e pode ser vista como a estrutura que cuida do acesso. As zonas publicas, por exemplo, são visíveis a todos os usuários, enquanto as zonas privadas só podem ser visualizadas por membros que possuem acesso a ela. Toda zona é constituída de pelo menos um *pod*.

3. *Pod*: é o componente que controla e seleciona recursos de hardware, ou seja, responsável por controlar os *clusters*. Um *Pod* pode ser pensando como um *rack* que contém um ou mais *clusters* controlados em uma mesma rede.

4. *Cluster*: o *cluster* é onde um grupo de *hosts*, que rodam com um hipervisor comum, possa ser agrupado. Cada *cluster* possui um armazenamento primário, o qual controla onde as máquinas virtuais estão hospedadas.

O modelo, ilustrado na Figura 3.4, desenvolvido para plataforma, permite cobrir todos os componentes básicos, prover todas as suas funcionalidades e cobrir todo o gerenciamento dos recursos. A interface de usuário fornecida pelo gerenciador é intuitiva e pode ser tanto por BUI ou por CLI.

O CloudStack fornece suporte para até seis hipervisores: KVM, Xen, Hyper-V, LXC, XenServer e vSphere[18]. Cada um suporta uma série de características, e cada *cluster* pode utilizar um diferente tipo de hipervisor.

O armazenamento possui a possibilidade de utilizar o sistema de arquivos NFS ou o protocolo iSCSI, ambos com suporte a tecnologia *Fiber Channel*. A elasticidade do CloudStack é pensada com escalonamento dinâmico, tanto horizontalmente quanto verticalmente. Se um *host* tiver a capacidade de escalonar mais recursos para uma VM ele o

[18] `http://www.vmware.com/br/products/vsphere-hypervidor`

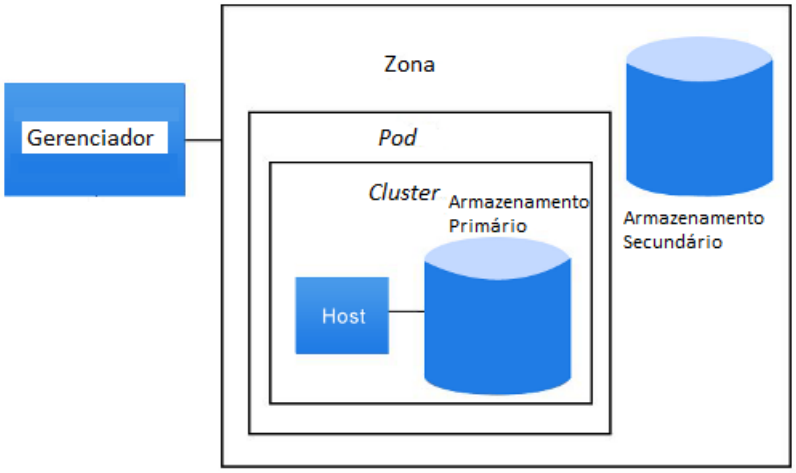

Figura 3.4: Arquitetura Hierárquica da Plataforma CloudStack, adaptado de [8].

fará (elasticidade vertical), caso contrário será observado se o *cluster* tem a capacidade de alocar mais recursos e, se possível, ocorrerá a migração da *VM* (elasticidade horizontal).

A autenticação pode contar com os serviços do protocolo LDAP e utilzar a linguagem SAML para novas métodos de autenticação. Outros serviços que o gerenciador provê, são serviços de redes como VLAN e *Flat Networking* para distribuir suas redes, o protocolo DHCP, que é utilizado para prover endereços IP automaticamente, e o serviço DNS para resolver nomes de domínios. Além disso, a plataforma também suporta VPN para configurar múltiplas zonas.

3.3.3 OpenNebula

O OpenNebula é uma plataforma de código aberto com a licença Apache 2.0. As atualizações são planejadas para cada três meses, mas esse tempo pode variar até seis meses, pois durante o período entre as atualizações ocorrem atualizações de manutenção, o que pode atrasar as atualizações definitivas. A documentação conta com vídeos, tutorias e textos abrangendo toda a visão geral da plataforma.

A interface de usuário pode ser por BUI ou CLI. A interface de API aceita diversas APIs, como, a Amazon EC2 e as APIs de linguagens de programação Ruby e java, entre outras.

O OpenNebula assume uma infraestrutura modular com quatro componentes básicos: *front-end*, rede, banco de dados e *host*. É uma maneira simples de divisão, como mostrada na Figura 3.5. O módulo de *front-end* cuida da interface com o usuário; o módulo de rede dos serviços de rede; o módulo de banco de dados do armazenamento; e o módulo de *Host* cuida da virtualização de máquinas.

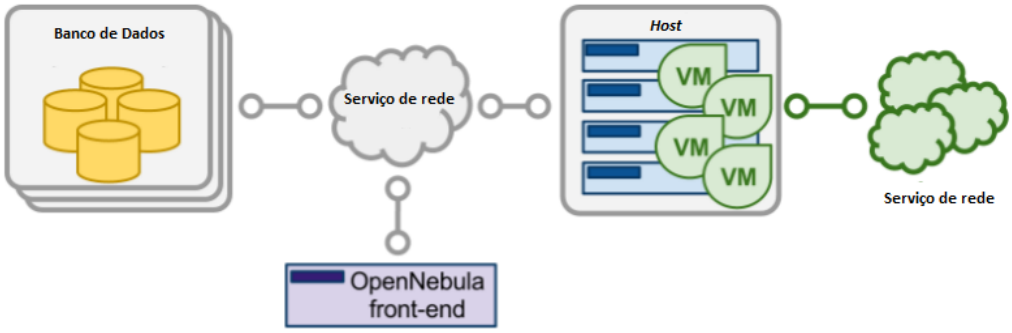

Figura 3.5: Arquitetura Modular da Plataforma OpenNebula, adaptado de [9].

Para o armazenamento, a plataforma possibilita o uso de NFS, *Fibre Channel* e/ou iSCSI. A elasticidade é horizontal, e possui o escalonamento com métricas que podem ser alteradas via comandos ou *scripts*, e quando uma ação de escalonamento inicia, o serviço entra no estado de escala, o qual irá instanciar ou desligar o número de VMs para chegar a sua nova cardinalidade.

O OpenNebula suporta trabalhar com três hipervisores: KVM, Xen e VMware. Entretanto, a plataforma foi primeiramente pensada e programada para interagir com o KVM. Assim, os demais hipervisores podem sofre perdas de funcionalidades se forem escolhidos para o projeto.

A autenticação conta com serviços tais como: X509, LDAP e SSH. A rede provê VLAN para a distribuição de rede, serviço de nome de domínio DNS e o protocolo DHCP para os endereços IPs.

3.3.4 Eucalyptus

O Eucalyptus possui a licença GPLv3 de código aberto, mas há algumas partes com licenciamento proprietário. As atualizações ocorrem entre seis a oito meses, dos quais a fase de produção ocorre nos primeiros seis meses. A documentação está toda disponível, mas é bastante resumida e pode ter difícil acesso a certas informações.

A interface de usuário pode precisar da ferramenta adicional gratuita Euca2ools[19], a qual fornece interface dos recursos através de linha de comandos, mas também há a interface BUI. O Eucalyptus possui interações com diversas APIs entre elas: Amazon EC2 e Amazon S3. Também há a possibilidade de escolha entre três hipervisores: KVM, Xen e VMware.

A plataforma possui uma arquitetura hierárquica com cinco componentes principais: *Cloud Controller*, *Scalable Object Storage*, *Cluster Controller*, *Storage Controller* e *Node Controller*. Como pode ser visto na Figura 3.6, os componentes podem ser divididos

[19] `https://github.com/eucalyptus/euca2ools`

em três camadas: Camada de nuvem, Camada de *cluster* e Camada de controle. Os componentes são descritos em detalhes a seguir:

1. *Cloud Controller* (CLC): é o ponto de entrada para os administradores e usuários finais. Ele consulta os outros componentes para informar e expor a gestão dos recursos virtualizados oferecendo uma interface.
2. *Scalable Object Storage* (SOS): permite a utilização da API Amazon S3 de armazenamento e/ou a solução básica do Eucalyptus para o armazenamento em pequenos projetos.
3. *Cluster Controller* (CC): possui a tarefa de gerenciar a execução de máquinas virtuais e disponibilizar dados para a interface oferecida pelo CLC.
4. *Storage Controller* (SC): também é responsável pelo gerenciamento de VMs, mas principalmente dos dados e do espaço em disco correspondente de cada VM.
5. *Node Controller* (NC): controla os nós que contêm as VMs e a virtualização de hardware.

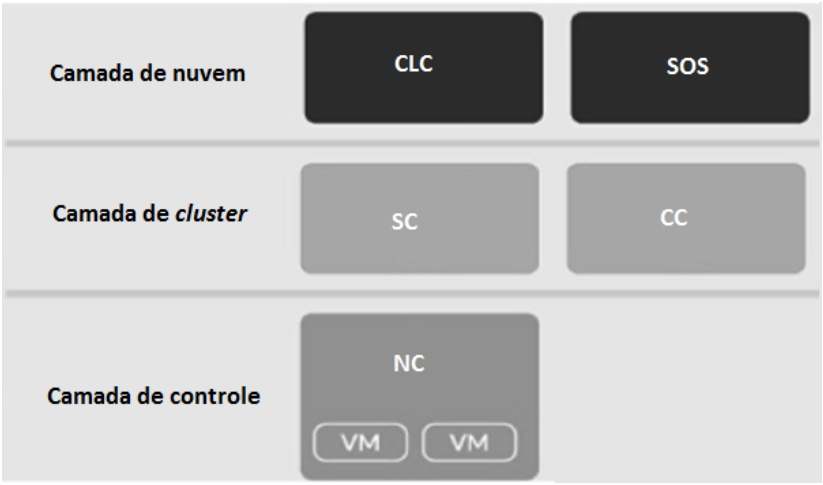

Figura 3.6: Arquitetura Hierárquica do Eucalyptus, adaptado de [10].

Para o armazenamento há apenas a possibilidade do uso de iSCSI. A elasticidade é horizontal e possibilita o escalonamento via comandos e/ou através de *scripts* para acelerar o processo. A configuração de rede pode utilzar VLAN ou *Flat Networking* para a distribuição das redes e DHCP para prover IPs automaticamente. Para autenticação o OpenNebula utiliza X509 e/ou LDAP.

3.3.5 Avaliação Comparativa das Plataformas

Todas as plataformas possibilitam aos usuários a criação e gerenciamento de nuvens privadas de IaaS. Entretanto, como mostrado nas subseções anteriores, cada plataforma possui diferentes considerações e características. Para a escolha da plataforma adequada para este projeto foi realizada uma comparação entre as principais plataformas. Nessa comparação foi considerado o ambiente acadêmico proposto para a implantação da nuvem privada.

Em relação à interface de usuário, todas as plataformas disponibilizam tanto interface através de linha de comando CLI, quanto por interface web BUI. As interfaces são bastante parecidas, mudando apenas questões estéticas e características de acordo com suas arquiteturas.

O OpenStack e o Eucalyptus oferecem suporte para diversas APIs e podem adaptar diversos serviços fornecidos por elas. Entretanto, todas as plataformas oferecem suporte para as APIs da Amazon (EC2 e S3), que são referências quando se fala em infraestrutura como serviço.

Quanto aos hipervisores, todas as plataformas possuem suporte para diferentes hipervisores, fornecendo flexibilidade para escolher o melhor gerenciamento de máquinas virtuais de acordo com a arquitetura da plataforma. As plataformas OpenStack e CloudStack se destacam, pois oferecem suporte a uma maior variedade de hipervisores.

A arquitetura é um conceito muito importante para a avaliação de uma plataforma. O OpenStack e o OpenNebula se caracterizam por apresentar uma arquitetura modular, enquanto o CloudStack e o Eucalyptus apresentam uma arquitetura hierárquica. A arquitetura do OpenStack chamou a atenção por sua proporção, pois possui dez subprojetos, cada qual com suas características e funcionalidades, mas para a implantação de uma nuvem todos devem ser estudados e analisados, elevando a complexidade do projeto. Porém a que mais se destacou foi a arquitetura do CloudStack, pela sua condensação, tornando o entendimento das funcionalidade da plataforma intuitivo e de fácil utilização.

Os serviços de rede VLAN, *Flat Networking* e DHCP são básicos, e por isso são fornecidos por todas as plataformas. Os serviços que mais chamaram a atenção foram o SDN, fornecido pelo OpenStack, e o suporte de VPN pelo CloudStack.

Todas as plataformas apresentaram algum tipo de protocolo para a transferência de dados e para o armazenamento em qualquer localização. Todavia, não foi encontrada nenhuma referência do Eucalyptus sobre a utilização do sistema de arquivos distribuídos NFS, característica básica apresentada pelas demais. Assim sendo, as demais plataformas possuem as mesmas características o suporte ao NFS a ao iSCSI. O suporte ao *Fiber Channel* para a interligação do armazenamento é interessante também, mas para isso

é necessário de um ambiente preparado para sua utilização, o qual não é o caso deste projeto.

Neste cenário, a elasticidade é uma característica essencial, e todas as plataformas tem suporte a esta característica. Contudo, o CloudStack destacou-se, pois possui é a única que apresentou suporte à elasticidade vertical.

Além disso, em relação à autenticação, o método comum a todos as plataformas é o LDAP, embora outros métodos sejam aceitos. O OpenStack é o que melhor apresentou soluções para autenticação, pois possui um módulo em sua arquitetura responsável só por operar problemas de identificação.

O CloudStack e o OpenStack são as plataformas que possuem uma comunidade grande de usuários, empresas envolvidas e desenvolvedores, por isso apresentam lançamentos e atualizações periódicas, com prazos determinados. O Eucalyptus demora mais em suas atualizações e o OpenNebula tenta se aproximar para estabelecer um prazo certo de sua atualização, mas ainda possuem prazos confusos para suas metas.

As plataformas OpenStack, CloudStack e OpenNebula possuem licença Apache 2.0 de código aberto, enquanto o Eucalyptus possui a licença de código aberto GPLv3, mas com alguns licenciamentos proprietários.

Além disso, a documentação foi essencial para avaliar as características e entender cada plataforma. O OpenStack, o CloudStack e o OpenNebula possuem uma documentação padronizada e bem estruturada, enquanto informações sobre o Eucalyptus foram difíceis de se entender.

Dessa forma, a Tabela 3.1 apresenta resumidamente todas as características que foram levados em consideração para avaliar as plataformas. Cada característica foi avaliada com uma nota de um a cinco (1 a 5) para representar o quão satisfatório e interessante é a característica da plataforma para este projeto, no qual cinco representa o maior nível de satisfação. As notas de um a cinco podem, respectivamente, serem comparadas com os seguintes graus de satisfação: nada satisfatório, pouco satisfatório, satisfatório, muito satisfatório e totalmente satisfatório.

Com base nas notas dadas às características, as plataformas mais interessantes foram o OpenStack e o CloudStack. A escolha entre ambas teve bastante peso na arquitetura, pois pelo fato de o CloudStack possuir uma arquitetura hierárquica e mais condensada será melhor para o ambiente a ser implantado neste projeto. Outro ponto importante na escolha feita é que o CloudStack é o único que apresenta elasticidade vertical dos recursos computacionais, sendo interessante para o ambiente de pesquisa do LABID[20]. Sendo assim, a plataforma CloudStack foi a escolhida para implantar a nuvem privada de IaaS no LABID da UnB.

[20] `http://www.cic.unb.br/pesquisa/laboratorios/`

Tabela 3.1: Tabela Comparativa das Plataformas de IaaS.

	OpenStack	**CloudStack**	**OpenNebula**	**Eucalyptus**
Interface de Usuário	Interface BUI e CLI **Nota:** 5	Interface BUI e CLI **Nota:** 5	Interface BUI e CLI **Nota:** 5	Interface BUI e CLI **Nota:** 5
Suporte de API	Amazon EC2, Amazon S3, CloudFiles REST, etc. **Nota:** 5	Amazon EC2 e Amazon S3 **Nota:** 4	OGF OCCI, Amazon EC2 e Amazon S3 **Nota:** 4	Amazon EC2, Amazon S3, IAM, etc. **Nota:** 5
Hipervisores	Suporte 8 hipervisores **Nota:** 5	Suporte 6 hipervisores **Nota:** 5	Suporte 3 hipervisores **Nota:** 3	Suporte 3 hipervisores **Nota:** 4
Arquitetura	Modular – Dividido em 10 subprojetos **Nota:** 3	Hierárquica – 4 componentes principais **Nota:** 5	Modular – 4 módulos **Nota:** 4	Hierárquica – 5 componentes principais **Nota:** 5
Rede	VLAN, Flat networking, DHCP e SDN **Nota:** 5	DNS, DHCP, VLAN, Flat networking e suporte para VPN **Nota:** 5	VLAN, DHCP, DNS **Nota:** 3	VLAN, Flat networking, DHCP **Nota:** 3
Armazenamento e Elasticidade	NFS, iSCSI, Fibre Channel. Elasticidade com escalonamento dinâmico horizontal **Nota:** 4	NFS, iSCSI, Fibre Channel. Elasticidade com escalonamento dinâmico (vertical e horizontal) **Nota:** 5	NFS, Fibre Channel, iSCSI. Elasticidade horizontal com escalonamento baseado em métricas **Nota:** 4	iSCSI. Elasticidade horizontal com possibilidade de escalonamento via comandos **Nota:** 2
Autenticação	Identity API, LDAP, Kerberos, X509 **Nota:** 4	LDAP, SAML 2.0 **Nota:** 3	X509, LDAP, SSH **Nota:** 3	X509, LDAP **Nota:** 3
Lançamento de Versões	6 meses **Nota:** 5	4 meses **Nota:** 5	Entre 3 a 8 meses **Nota:** 4	6 a 8 meses **Nota:** 3
Licença	Apache 2.0 License **Nota:** 5	Apache 2.0 License **Nota:** 5	Apache 2.0 License **Nota:** 5	GPLv3 com licenciamento proprietário **Nota:** 2
Documentação	**Nota:** 5	**Nota:** 5	**Nota:** 5	**Nota:** 2
Total	46 pontos	47 pontos	40 pontos	37 pontos

3.4 Considerações Finais

Neste capítulo foi apresentado um dos principais serviços que envolve nuvens de IaaS, que é a virtualização de hardware. Depois foi realizada uma pesquisa visando conhecer as principais soluções e as principais características que envolvem a implantação de nuvens de IaaS na Seção 3.2. Por último, na Seção 3.3, foram avaliadas as principais plataformas de código aberto disponíveis atualmente.

A plataforma escolhida para a implantação da nuvem privada no LABID foi a CloudStack. Assim, o Capítulo 4 apresentará a ambientação deste projeto, assim como, os passos necessários para realizar a implantação de uma nuvem privada de IaaS.

Capítulo 4

Implantação da Nuvem Privada

Este capítulo apresenta a implantação da nuvem privada de IaaS no LABID. A implantação foi realizada por meio da ferramenta CloudStack, escolhida através da avaliação das plataformas apresentadas no Capítulo 3.

A Seção 4.1 abordará o cenário computacional em que a nuvem está inserida. A Seção 4.2 apresentará a arquitetura definida para a implantação. Em seguida, a Seção 4.3 mostrará as configurações e as instalações necessárias, enquanto a Seção 4.4 trará a interação com o gerenciador de nuvem e os últimos passos para completar a implantação da nuvem privada de IaaS.

4.1 Ambiente

Para a implantação da nuvem privada de infraestrutura foram disponibilizados recursos do Laboratório de Bioinformática e Dados (LABID)[1]. Esse laboratório concentra as pesquisa de *big data*, de sistemas distrubuídos e Bioinformática. Um dos projetos realizados neste laboratório é o projeto BioNimbuZ, proposta originalmente por Saldanha [13].

O BioNimbuZ é uma plataforma de execução de *workflows* de bioinformática que utiliza uma infraestrutura provida por uma federação de nuvens híbridas, e que tem sido aprimorada constantemente em outros trabalhos [54] [55] [56]. O projeto consiste em montar um ambiente de federação de nuvens, no qual cada provedor de nuvem é capaz de ampliar, de forma transparente, a sua própria quantidade de recursos, requerendo maior poder computacional e/ou capacidade de armazenamento para outras nuvens.

A plataforma BioNimbuZ permite a federação de nuvens de diversos tipos, tanto privadas quanto públicas. Dessa forma, cada provedor pode manter suas características e políticas internas, e oferecer ao usuário transparência e ilusão de recursos virtualmente

[1] `http://www.cic.unb.br/pesquisa/laboratorios/`

ilimitados. Assim, o usuário pode usufruir de diversos serviços sem ter que gerenciar a infraestrutura que está sendo utilizada. Entretanto, não existe nenhuma nuvem de infraestrutura no LABID para apoiar o desenvolvimento da plataforma BioNimbuZ.

Nesse contexto, a motivação deste trabalho é implantar uma nuvem privada de IaaS no LABID para servir de apoio a projetos de pesquisa, como por exemplo, BioNimbuZ. As máquinas disponibilizadas para projetar a arquitetura da implantação podem ser vistas na Tabela 4.1. Além dessas máquinas, também foi disposto um *switch* modelo Summit24e2.

Tabela 4.1: Máquinas Utilizadas para a Implantação da Nuvem Privada de IaaS.

Apelido	**Modelo**	**Processador**	**Nº de Processadores**	**Frequência**	**RAM**	**Espaço de disco**
Lab_01	hp compaq 8200 ESFF	Core i5	4	3,10 GHz	8 GB	500 GB
Lab_02	hp compaq 8200 ESFF	Core i5	4	3,10 GHz	8 GB	500 GB
Lab_03	IBM x3550	Xeon	2	1,60 GHz	8 GB	136 GB

4.2 Arquitetura

Como visto na Seção 3.3.2, a plataforma CloudStack possui quatro componentes básicos em sua arquitetura: Gerenciador, Zona, *Pod* e *Cluster*. O modelo no qual a plataforma foi disposta permite que a partir de um gerenciador todos os demais componentes possam ser controlados. Então, com o ambiente disponibilizado no LABID para a construção da nuvem de infraestrutura, a seguinte estratégia foi utilizada:

1. Utilizar a máquina Lab_03 como o gerenciador da nuvem de IaaS, servidor de dados (com o armazenamento primário e secundário) e servir como um roteador;
2. Adotar as máquinas Lab_01 e Lab_02 como *hosts* para formarem um *cluster*, pois todos os *hosts* em um *cluster* devem ser homogêneos;
3. Por fim, utilizar o *switch* para interligar as máquias à rede;

A Figura 4.1 ilustra a estratégia proposta para a arquitetura da nuvem privada de IaaS montada neste trabalho, com o CloudStack. Assim, é possível notar que na máquina Lab_03 foi instalado o gerenciador, atribuído um endereço IP público provido pela UnB e instalado o servidor de dados; As máquinas Lab_01 e Lab_02 foram ligadas a rede interna pelo *switch* para formarem o *cluster* da nuvem privada.

O sistema operacional escolhido para realizar a implantação da nuvem foi o CentOS[2]. Ele foi o sistema escolhido pelo fato de ser considerado um dos melhores sistemas para

[2]`https://www.centos.org/`

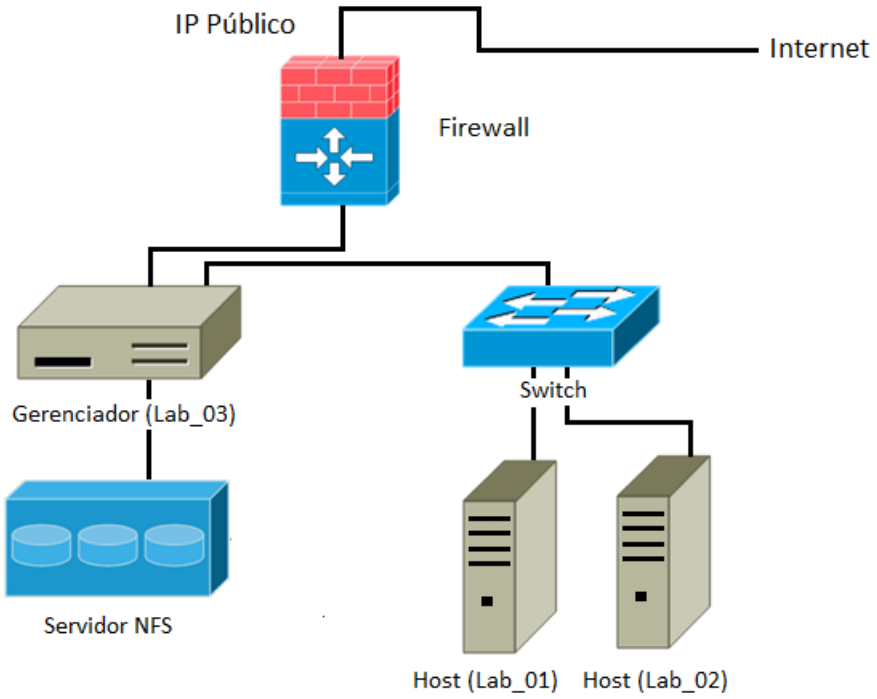

Figura 4.1: Arquitetura Utilizada para Implantação da Nuvem Privada.

servidores, e é amplamente utilizado para a implantação das plataformas de IaaS. A versão utilizada foi CentOS 6.7 x86_64_minimal.

Como dito anteriormente, na Seção 3.3.2, o CloudStack suporta diferentes hipervisores, que podem ser usados simultaneamente. Entretanto, como o projeto conta com apenas um *cluster*, formado pelas máquinas Lab_01 e Lab_02, foi utilizado somente o hipervisor KVM[3].

4.3 Configuração e Instalação

Após a definição da arquitetura a ser adotada para a infraestrutura, faz-se necessário configurar o ambiente para que a nuvem possa ser instalada com a plataforma CloudStack. A descrição dos procedimentos necessários serão descritos nas subseções subsequentes. O passo-a-passo realizado para os procedimentos foi:

1. Configurar a rede entre as máquinas da nuvem;
2. Ajustar o SELinux e o protocolo NTP;
3. Ajustar os repositórios com as versões do CloudStack;

[3] `http://www.linux-kvm.org/page/Main_Page`

4. Definir e configurar o sistema de arquivos para o servidor de arquivos (armazenamento primário e secundário);
5. Instalar o gerenciador de nuvem CloudStack;
6. Configurar os *hosts* com o hipervisor KVM para a integração no *cluster*;

4.3.1 Passo 1 - Configuração de Rede

A primeira máquina configurada foi a Lab_03, pois, como definido na arquitetura da Seção 4.2, foi disposta para ser o roteador entre a rede pública fornecida pela UnB e a rede interna da nuvem. Para isso, foram configuradas duas placas de rede, uma com IP público e outra com IP privado. Os passos utilizados para realizar essa configuração foram:

1. Abrir os arquivos das placas de rede eth1 e eth2 da máquina Lab_03;
2. Modificar os arquivos, como exemplificado pela Figura 4.2. Os campos adicionados foram: GATEWAY, NETMASK, BOOTPROTO, ONBOOT, DNS1E DNS2. O DNS1 e DNS2 são endereços IP para servidores da própria UnB;
3. Para o campo IPADD, o IP público 164.41.209.100 foi utilizado para a placa de rede eth1, e o IP privado 192.168.0.11 para a placa de rede eth2;

```
DEVICE=eth1
HWADDR=00:14:5E:17:84:AC
TYPE=Ethernet
#VID=93a95715-f604-4a21-8013-8aa300ca4f02
ONBOOT=yes
NM_CONTROLLED=mo
BOOTPROTO=static
IPADDR=164.41.209.100
NETMASK=255.255.255.0
GATEWAY=164.41.209.1
DNS1=164.41.211.34
DNS2=164.41.101.11
```

Figura 4.2: Configuração Final da Placa de Rede Eth1 da Máquina Lab_03.

Em seguida, foi necessário configurar as placas de rede eth0 das máquinas Lab_01 e Lab_02. Como visto na Seção 4.2, estas máquinas foram destinadas para serem os *hosts* da nuvem, ou seja, são nelas que serão instanciadas as VMs da nuvem. Por isso, a configuração de cada placa necessitou da utilização da técnica *Bridged Networking* para permitir que as interfaces de rede das VMs pudessem utilizar a placa de rede física do *host* em que está alocada, isto é, permitir que as VMs sejam vistas como um *host* físico para o resto da rede. Os passos utilizados para realizar as configurações de cada máquina foram:

1. Abrir os arquivos das placas de rede eth0 das máquinas Lab_01 e Lab_02;
2. Modificar o arquivo como exemplificado pela Figura 4.3 para permitir a *bridge* denominada cloudbr0;
3. Criar o arquivo de configuração da *bridge* cloudbr0 (ifcfg-cloudbr0) para poder configurá-la, conforme a Figura 4.3. Para a máquina Lab_01 foi utilizado o IP privado 192.168.0.16 e para a máquina Lab_02 o IP privado 192.168.0.17;

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
BRIDGE=cloudbr0
```

Figura 4.3: Configuração Final da Placa de Rede Eth0 da Máquina Lab_01.

```
DEVICE=cloudbr0
TYPE=Bridge
ONBOOT=yes
IPADDR=192.168.0.16
PREFIX=24
GATEWAY=192.168.0.254
DNS1=164.41.211.34
DNS2=164.41.101.11
BOOTPROTO=none
IPV6_AUTOCINF=no
IPV6INIT=no
DELAY=5
STP=yes
```

Figura 4.4: Configuração Final da *Bridge* Cloudbr0 na Máquina Lab_01.

4.3.2 Passo 2 - SELinux e NTP

O passo seguinte para para a implantação da nuvem privada de IaaS no LABID foi a configuração do SELinux (*Security-Enhanced Linux*) em cada uma das máquinas. O SELinux é padrão do sistema operacional escolhido, CentOS, com o objetivo é prover maior segurança a arquivos, diretórios, processos e sistema de arquivos. O SELinux trabalha, basicamente, com três modos:

1. Modo *enforcing*, no qual todas as regras são aplicadas, e são gerados *logs* de todas as operações;

2. Modo *Permissive*, no qual o SELinux continua habilitado, mas com suas regras desabilitadas. Ou seja, o sistema não é protegido pelo SELinux, mas tem o propósito de registrar os mesmos *logs* do modo *enforcing*, permitindo que o administrador tenha uma noção do estado do sistema;

3. Modo *disable*, no qual as políticas e os *logs* são desabilitados por completo;

Para o funcionamento do CloudStack, o SELinux deve estar no modo *permissive* nas máquinas onde estão o gerenciador e os *hosts*, pois o modo *enforcing* pode ocasionar o bloqueio de alguns serviços de rede para a nuvem. Para isso, deve-se modicar o arquivo de configuração SELinux (*/etc/selinux/config*), como mostrado na Figura 4.5.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Figura 4.5: Arquivo de Configuração SELinux.

Em seguida, fez-se necessário configurar e ativar o protocolo NTP (*Network Time Protocol*) [57] em todas as máquinas, para a sincronização do conjunto de relógios e manter a hora certa e sincronizada entre elas.

4.3.3 Passo 3 - Repositório

A plataforma CloudStack trabalha com repositórios disponíveis pela Internet com suas versões. Então, é necessário configurar todas as máquinas (Lab_03, Lab_02 e Lab_01) para que os mesmos possam poder usar o repositório da versão a ser instalada. Para isso, deve-se criar o arquivo */etc/yum.repos.d/cloudstack.repo*, e configurá-lo como mostrado na Figura 4.6.

```
[cloudstack]
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/centos/6/4.5/
enabled=1
gpgcheck=0
```

Figura 4.6: Configuração do Arquivo de Repositório para a Plataforma CloudStack.

4.3.4 Passo 4 - Sistema de Arquivos

Como visto na arquitetura mostrada na Figura 4.1, precisa-se de um servidor de armazenamento. Assim sendo, faz-se necessário configurar o sistema de arquivos para gerenciar o armazenamento primário e o secundário.

Na nuvem, o papel do armazenamento primário está ligado aos *clusters*, sua função é guardar informações e dados das máquinas virtuais. O armazenamento secundário tem a função de guardar *templates* de configurações, imagens de sistemas operacionais e, eventualmente, dados para serem transferidos entre *clusters*.

O CloudStack trabalha tanto com o iSCSI (*Internet Small Computer System Interface*) [43] quanto com o NFS (*Network File System*) [44]. Radkov *et al.* [58] defendem que a principal diferença entre os métodos está no nível de abstração. O NFS utiliza a técnica de abstração dos dados em arquivos, e de organização de arquivos em diretórios entre computadores conectados em rede, enquanto o iSCSI gerencia os dados através de blocos de disco.

Para este projeto foi escolhido o NFS, pois pode ser utilizado para ambos os armazenamentos (primário e secundário). Além disso, o NFS possui uma melhor compatibilidade com o hipervisor KVM escolhido para este projeto.

Para a configuração do servidor NFS, na máquina Lab_03, foram criados dois novos diretórios: *primary* e *secundary*. Em seguida, foi instalado o NFS-*utils* que contém as ferramentas necessárias para o servidor.

Depois, foram configuradas algumas permissões para os diretórios. Para isso, foi mudado o arquivo */etc/exports*, e adicionado o conteúdo conforme apresentado na Figura 4.7. Em seguida, algumas configurações do NFS foram alteradas no arquivo */etc/sysconfig/nfs*, conforme os valores apresentados na Figura 4.8.

```
/secondary *(rw,async,no_root_squash,no_subtree_check)
/primary *(rw,async,no_root_squash,no_subtree_check)
```

Figura 4.7: Permissões NFS para o Servidor de Arquivos.

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
RQUOTAD_PORT=875
STATD_PORT=662
STATD_OUTGOING_PORT=2020
```

Figura 4.8: Configuração NFS do Servidor de Arquivos.

Para finalizar a configuração do servidor NFS, foi preciso utilizar a ferramenta *iptables* para permitir que o *firewall* aceitasse as configurações das portas, conforme mostrado na Figura 4.9.

```
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p udp --dport 111 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 111 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 2049 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 32803 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p udp --dport 32769 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 892 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p udp --dport 892 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 875 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p udp --dport 875 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p tcp --dport 662 -j ACCEPT
-A INPUT -s 192.168.0.0/24 -m state --state NEW -p udp --dport 662 -j ACCEPT
```

Figura 4.9: Configuração da *Firewall* para o Servidor de Arquivos.

4.3.5 Passo 5 - Servidor Gerenciador

O primeiro passo realizado, para a configuração do servidor de gerência do CloudStack na máquina Lab_03, foi a instalação do MySQL *Server*[4]. Ele foi necessário para o criação do banco de dados do gerenciador da nuvem. Depois, o arquivo */etc/my.cnf* de configuração do MySQL foi editado conforme a Figura 4.10, para atender os padrões do gerenciado, e o MySQL configurado para sempre inciar quando a máquina for ligada.

```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
```

Figura 4.10: Configuração do MySQL.

Depois de configurado o banco de dados, com o sistema gerenciador de banco de dados MySQL, o gerenciador é de simples instalação. Há apenas três comandos a serem executados: um para a instalação propriamente dita; outro para configurá-lo com o banco de dados; e outro para finalizar a configuração do gerenciador.

4.3.6 Passo 6 - Hipervisor

Seguindo a lógica da arquitetura, apresentada na Seção 4.2, as máquinas Lab_01 e Lab_02 foram escolhidas para atuar como *hosts* de um *cluster* por meio do hipervisor

[4] https://www.mysql.com/

KVM. As máquinas não estavam programadas para usar o KVM, então foram alteradas algumas configurações em suas BIOS.

O primeiro passo realizado foi a instalação do CloudStack *Agent*, que serve para conectar o gerenciador com os *hosts*, e gerenciar a instanciação de VMs no hipervisor KVM. Para o gerenciamento, o CloudStack *Agent* utiliza a API libvirt[5] para suportar a virtualização disponibilizada pelo hipervisor KVM. Por isso, foi necessário editar o arquivo de configuração do libvirt (*/etc/libvirt/libvirtd.conf*), API de virtualização para hipervisores, como mostrado na Figura 4.11. Por fim, o arquivo */etc/sysconfig/libvirtd* foi modificado para que a API pudesse ser identificada pelo sistema conforme a Figura 4.12.

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16059"
auth_tcp = "none"
mdns_adv = 0
```

Figura 4.11: Valores Adicionados às Configurações do Libvirt.

```
LIBVIRTD_ARGS="--listen"
```

Figura 4.12: Comando para Identificação do Libvirt.

4.4 Gerenciador de Nuvem CloudStack

Após todas as configurações realizadas na Seção 4.3, é possível utilizar o Cliente (*Client* em inglês) do CloudStack, no qual é possível acessar remotamente os serviços oferecidos pela plataforma. O Cliente é disponível através de uma interface web via *browser* (interface BUI).

O acesso é realizado pelo protocolo HTTP que disponibiliza o acesso remoto ao servidor gerenciador da nuvem via porta 8080. Para acessá-lo foi utilizado o endereço IP público do servidor gerenciador (164.41.209.100). O URL padrão para nuvens implantadas com a plataforma CloudStack é http://<endereço-ip-gerenciador>:8080/client

Para acessar a interface do servidor gerenciador da nuvem, foi necessário logar na nuvem. A configuração básica de identificação da nuvem utiliza o protocolo LDAP, o qual possui um servidor embutido ao gerenciador, com cada usuário ligado a uma senha. A tela de *login* exige que o usuário entre com:

1. Nome de usuário;

[5] `https://libvirt.org/`

2. Senha associada a identificação do usuário;

3. Caminho do domínio: este campo é usado quando o gerenciador não for implantado por um usuário *root*, e então, deve ser informado o caminho correspondente a instalação. Caso contrário o campo pode ser deixado em branco;

4. Língua: o gerenciador da nuvem pode ser acessado em dezesseis línguas diferentes, entre elas: inglês, português (Brasil), francês, chines, koreano, etc;

Além disso, a plataforma permite criar dois tipos de usuários para a nuvem, o usuário comum e o usuários administrador. O usuário comum pode acessar os recursos da nuvem destinados a eles como: instância de máquinas virtuais, *templates*, sistemas operacionais, projetos, espaço de armazenamento, etc. Já o usuário administrador está um nível acima dos usuários comuns. Por isso, tudo que for criado por usuários comum pode ser acessado e monitorado pelo administrador. Além disso, podem gerenciar e monitorar toda a infraestrutura e os recursos da nuvem, tais como, rede, armazenamento, infraestrutura, usuários, eventos, projetos, etc. A Figura 4.13 mostra a interface do painel de instrumentos disponíveis para os administradores e usuários comuns.

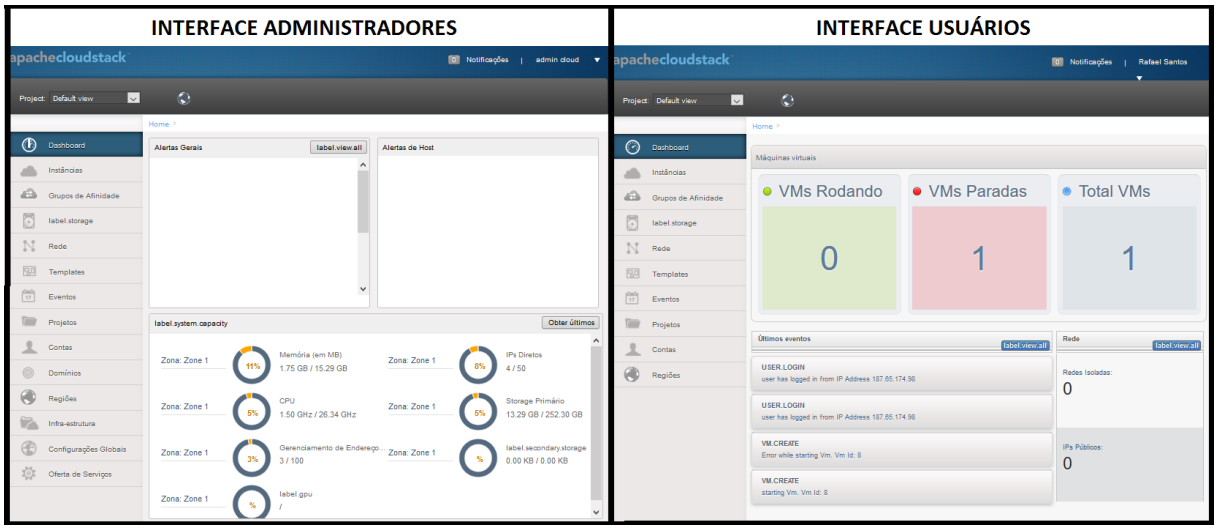

Figura 4.13: À Esquerda a Interface de Usuário Administradores e à Direita a interface de Usuário Comum.

Por padrão, o primeiro usuário criado para utilização do servidor gerenciador da nuvem é o administrador. Após o primeiro acesso, é necessário configurar os elementos Zona e *Pod* da arquitetura do CloudStack, apresentados na Seção 3.3.2.

A zona foi configurada com os endereços do servidores DNS da própria UnB, os quais são: 164.41.211.34 e 164.41.101.11. Além disso, a Zona foi configurada para que cada VM possuísse apenas um único endereço IP, por isso foi configurado o DNS interno 192.168.0.250. O *Pod* foi configurado com alcance de endereços IPs de 192.168.0.101

até 192.168.0.200. Também foi adicionado o *gateway* 192.168.0.1 com a máscara de subrede 255.255.255.0. Além disso, foi destinado o intervalo de IPs para máquinas virtuais da nuvem de 192.168.0.51 até 192.168.0.100.

Após toda as configurações e instalações necessárias para a implantação da nuvem privada de IaaS no LABID é possível ver através da interface BUI do gerenciador todos os componentes da infraestrutura final da nuvem, como mostrado na Figura 4.14.

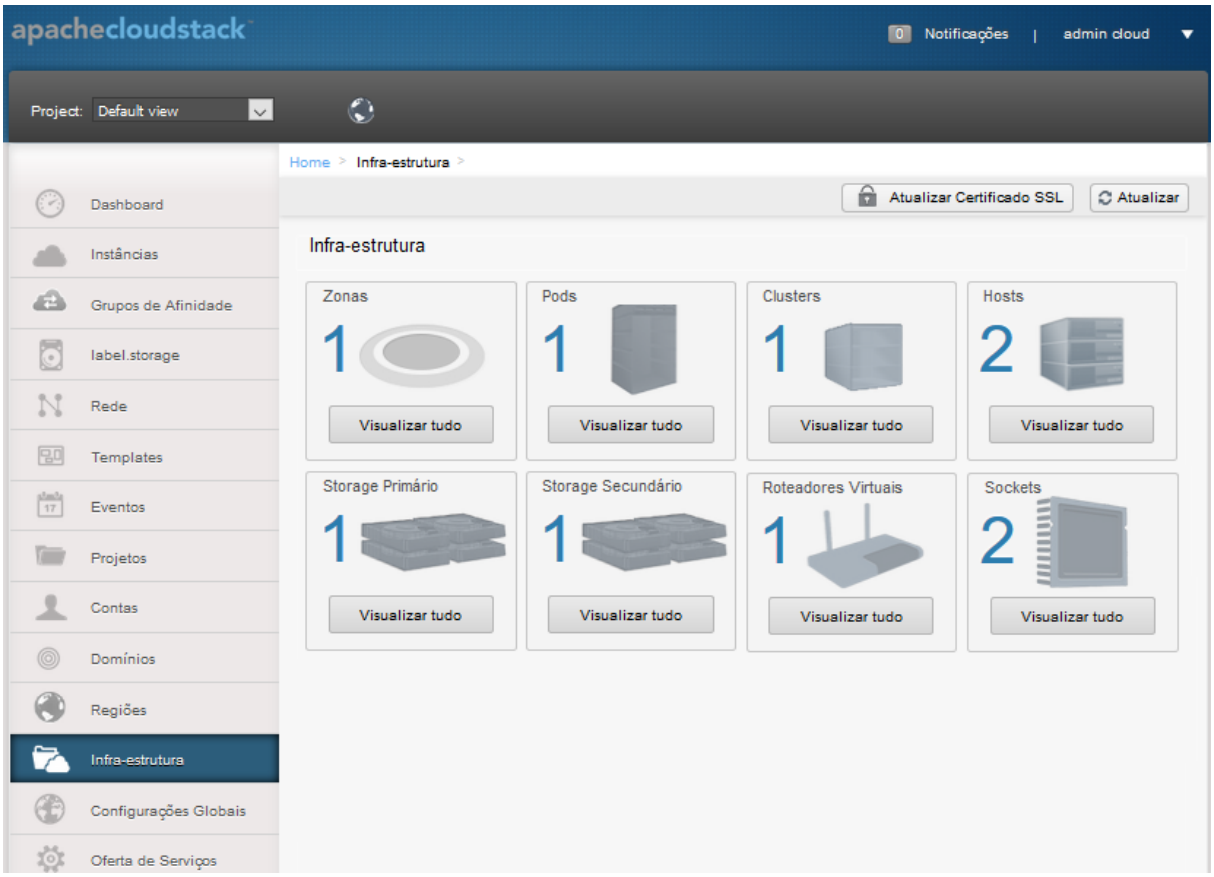

Figura 4.14: Infraestrutura Final da Nuvem Privada do LABID.

4.5 Considerações Finais

Neste capítulo foi apresentado os passos realizados para a implantação da nuvem privada de IaaS. Para isso, foi exposto o ambiente para qual a nuvem foi pensada, o qual é laboratório LABID da UnB. Além disso, foi indicado a arquitetura proposta para a implantação com a utilização dos recursos disponíveis.

O Capítulo 5 trará os testes realizados sobre a nuvem implantada e apresentada neste capítulo para o LABID.

Capítulo 5

Testes e Resultados Obtidos

Neste capítulo serão mostrados os testes realizados para observar a integração, o comportamento e o desempenho da nuvem privada de IaaS implantada no LABID. Para isso, a Seção 5.1 mostra o teste realizado para analisar a integração da nuvem privada com a plataforma de Bioinformática BioNimbuZ. A Seção 5.2 aborda os testes realizados para analisar o comportamento e a performance de algumas características chaves da nuvem implantada. A Seção 5.3 apresenta as considerações finais sobre os testes realizados na nuvem privada de IaaS implantada no LABID.

5.1 Teste de Integração

O principal objetivo deste teste foi garantir a integração entre a nuvem privada de IaaS implantada e a plataforma de Bioinformática BioNimbuZ. Para isso, foram executados testes com um *workflow* de Bioinformática real em uma VM preparada na nuvem implantada no LABID.

5.1.1 Arquitetura do BioNimbuZ

Para a realização do teste de integração foi necessário entender como funciona a arquitetura do BioNimbuZ. Ramos [56] apresenta a proposta da atual arquitetura utilizada pela plataforma, a qual possui quatro camadas: camada de aplicação, de integração, de núcleo e de infraestrutura. A nuvem privada implantada no LABID oferece IaaS para a camada de infraestrutura do BioNimbuZ, como mostra a Figura 5.1. As camadas da plataforma serão descritas a seguir:

1. A camada de Aplicação é composta pela aplicação web, a qual fornece a interface para o sistema gerenciador de *workflows* de Bioinformática, e utiliza a camada de integração para enviar requisições e receber respostas da camada de núcleo;

2. A Camada de Integração é responsável por integrar as Camadas de Aplicação e de Núcleo. Sua função é realizar a troca de mensagens entre essas camadas utilizando *webservices*, disparando requisições da Camada de Aplicação para o Núcleo e recebendo respostas do Núcleo para a Aplicação;

3. A camada de Núcleo é responsável pelo armazenamento de arquivos e metadados, escalonamento de *jobs*, execução de *workflows*, descobrimento dos provedores e o gerenciamento de possíveis falhas;

4. A camada de Infraestrutura é composta pelos provedores de IaaS utilizados pelo BioNimbuZ, e que compõem sua federação de nuvens.

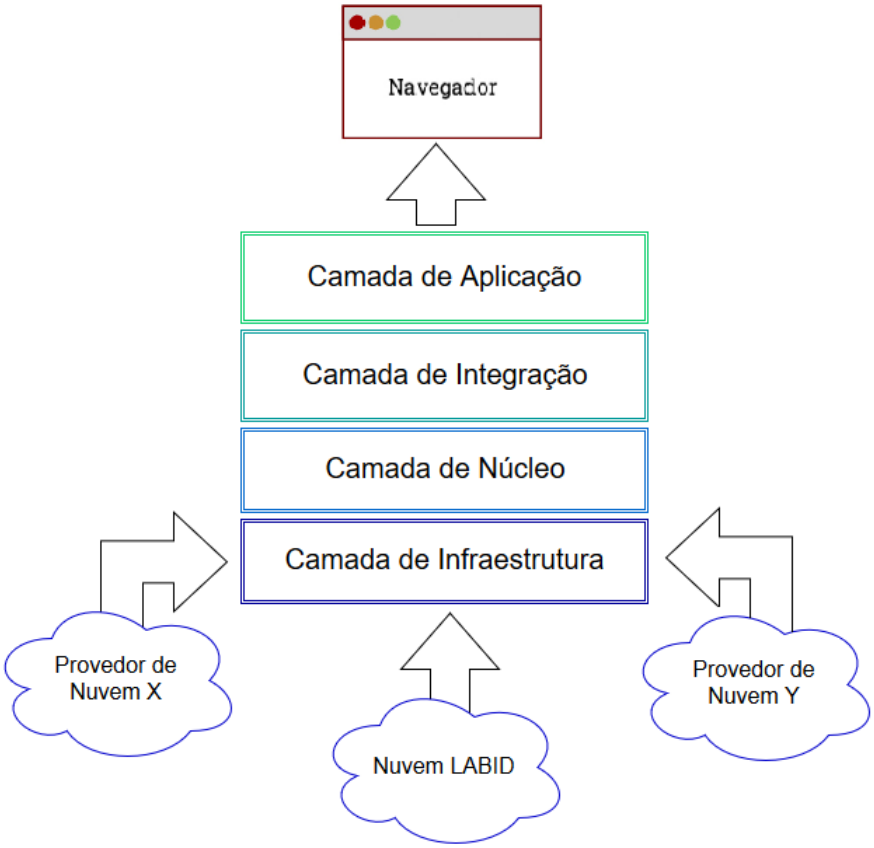

Figura 5.1: Arquitetura da Plataforma BioNimbuZ.

5.1.2 Ambiente

Para a realização do teste de integração foi utilizada uma VM instanciada na nuvem privada com 6 GB de memória RAM, 4 núcleos de CPU com 2 GHz de frequência e um disco rígido de 10 GB de capacidade. Também foi utilizado o sistema operacional CentOS

5.5 x86_64[1] para a VM. Além disso, foi utilizada uma máquina de uso pessoal, à qual estava conectada a nuvem implantada, para poder utilizar a interface *web* disponibilizada pela camada de aplicação da plataforma BioNimbuZ. A Figura 5.2 apresenta esse ambiente.

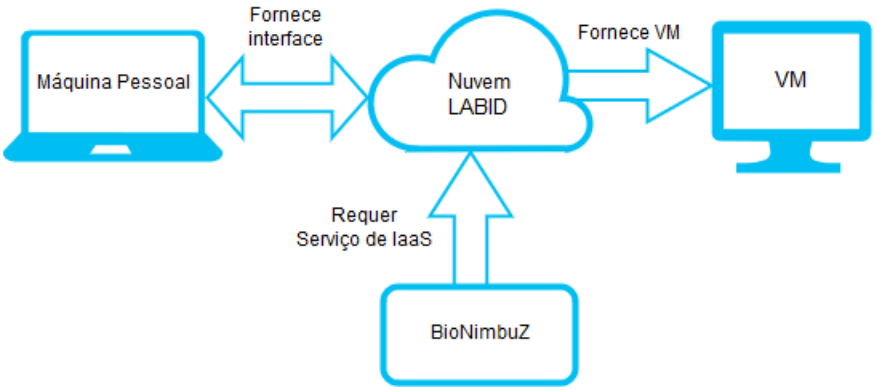

Figura 5.2: Ambiente para Teste de Integração.

O *workflow* de Bioinformática utilizado tem o objetivo de identificar genes diferencialmente expressos em células humanas cancerosas do rim e do fígado [59] [60], com fragmentos gerados pelo sequenciador Illumina [2]. Além disso, foram utilizados como entrada os 24 cromossomos do genoma de referência humano (hg19), os quais foram obtidos do banco de dados NCBI (*National Center for Biotechnology Information*)[3]. Também, foram utilizadas as ferramentas Bowtie[4] e Sam2Bed[5] para a execução do *workflow*. A ferramenta Bowtie é utilizada para mapear os fragmentos nos 24 cromossomos humanos, enquanto a ferramenta Sam2Bed é utilizada para conversão de *scripts*, no caso, para converter o arquivo de saída do Bowtie (arquivo .SAM) para o formato .BED. A Figura 5.3 representa o *workflow* utilizado para a realização do teste.

5.1.3 Descrição do Teste Realizado

Para a realização do teste de integração, primeiramente, foi instanciada uma VM na nuvem privada do LABID, e configurada com os procedimentos necessários para a instalação da plataforma BioNimbuZ [61]. Depois, na máquina pessoal foi instalado o BioNimbuZ *Client* [62]. Além disso, foi preciso configurar o *firewall* da nuvem para permitir que a aplicação BioNimbuZ pudesse se comunicar via protocolo TCP pela porta 8181. As-

[1] `https://www.centos.org/`
[2] `http://www.illumina.com`
[3] `http://www.ncbi.nlm.nih.gov`
[4] `http://bowtie-bio.sourceforge.net/index.shtml`
[5] http://bedops.readthedocs.io/en/latest/content/reference/file-management/conversion/sam2bed.html

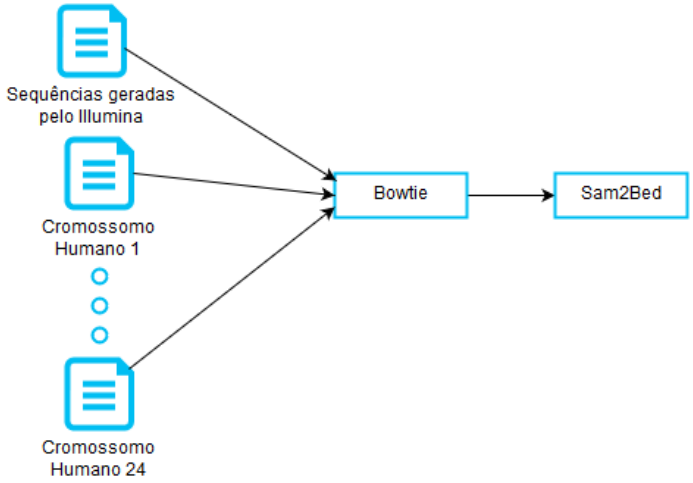

Figura 5.3: *Workflow* Utilizado para Teste.

sim, a VM ficou responsável por todo o processamento do *workflow* proposto, enquanto a máquina pessoal por acessar a interface que a camada de aplicação do BioNimbuZ oferece.

A partir do *Client* foi possível acessar a interface *web* e executar o *workflow* proposto. Primeiramente, foi realizado o *login* na plataforma, carregado o arquivo de entrada gerado pelo sequenciador Illumina para comparar com o arquivo de referência dos cromossomos humanos. Depois, foi configurado o *workflow* com duas etapas. A primeira etapa utilizou da ferramenta Bowtie para mapear os fragmentos nos 24 cromossomos humanos, e a segunda etapa utilizou a ferramenta Sam2Bed para converter o arquivo de saída do Bowtie para .BED.

Após a execução do *workflow*, a tela de sucesso foi obtida. A Figura 5.4 mostra a confirmação do teste realizado, na qual, mostra que o *workflow* chegou ao servidor do BioNimbuZ, foi iniciado e escalonado, foram executas os *jobs* necessários, e o *workflow* foi finalizado. Com isso, o teste de integração foi validado, demostrando que a nuvem privada de IaaS implantada no LABID é capaz de atender um dos projetos de pesquisas realizadas no LABID sobre computação em nuvem, o BioNimbuZ.

5.2 Teste de Comportamento e de Desempenho

O objetivo principal do teste de comportamento e de desempenho, da nuvem privada implantada no LABID, é avaliar o comportamento de algumas características chaves no ambiente de máquinas virtuais. Paradowski *et al.* [63] e AL Mukhtar *et al.* [64] realizaram estudos envolvendo uma série de testes para nuvens implantadas com a plataforma CloudStack. Neste contexto, foi pensado em cinco testes principais. Os respectivos testes e seus objetivos são:

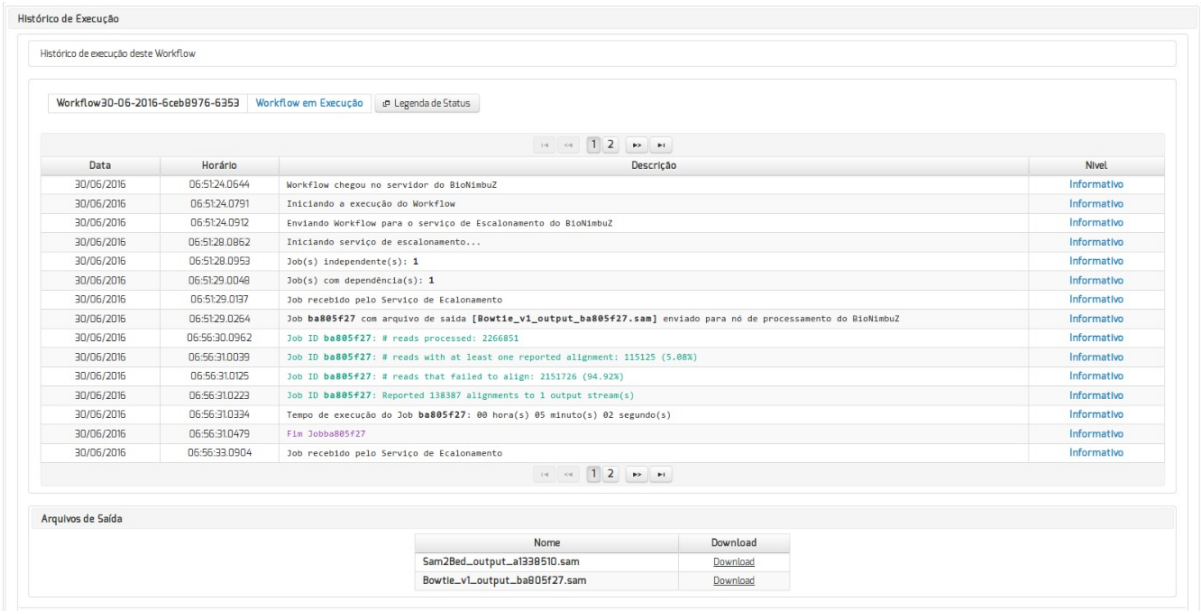

Figura 5.4: Tela de Sucesso ao Executar o *Workflow* de Bioinformática na Nuvem Implantada.

1. Provisionamento de VM: o objetivo desde teste é analisar fatores que possam interferir no tempo de criação de VMs;

2. Escalonamento de recursos: o propósito principal é testar como funciona o escalonamento de recursos para VMs em uma nuvem implantada com o CloudStack, o qual foi um dos pontos fundamentais para a escolha da plataforma, como visto na Seção 3.3.5;

3. Perfomance: A finalidade deste teste é avaliar o quão próximo o desempenho de uma VM está do seu desempenho teórico, ou seja, desempenho que uma máquina com as mesma configurações deveria apresentar;

4. Isolamento de VM: este teste está relacionado com a avaliação da performance. O objetivo é observar o desempenho de um ambiente com várias VMs em uma nuvem ao mesmo tempo.

5. Operações de disco: O intuito é observar o comportamento de operações de disco rígido, e analisar o comportamento do sistema de arquivos NFS.

5.2.1 Ambiente Escolhido

Para realização dos testes apresentados nesta seção, foi utilizado o sistema operacional CentOS 5.5 x86_64 para todas as VMs. O sistema operacional foi disponível através de

um *template* de sistema operacional para facilitar e acelerar os testes, ou seja, não é preciso instalar e configurar o sistema operacional a cada teste.

A nuvem de IaaS, implantada com a plataforma CloudStack, permite a total customização das configurações de VMs (dentro dos limites da nuvem). Então, para a realização dos testes foram escolhidas quatro tipos de configurações para testar as VMs. A Tabela 5.1 representa os diferentes tipos de configurações utilizadas para avaliar o desempenho e o comportamento da nuvem privada implantada.

Tabela 5.1: Tabela com a Configuração dos Quatros Tipos de VMs Utilizadas para Testes.

Tipo	**RAM**	**Nº de núcleos**	**Frequência**
Pequena	1 GB	1	2 GHz
Média	2 GB	2	2 GHz
Grande	4 GB	2	2 GHz
Extra grande	6 GB	2	2 GHz

Para realização de alguns testes foram utilizados *benchmarks*, que é uma forma de analisar produtos e/ou serviços a fim de comparar uma determinada atividade e realizar um estudo de desempenho. Para a realização dos testes de performance e isolaneto de VMs foi utilizado o Linpack [65]; enquanto para os testes de operações de disco foi utilizado o Bonnie++ [66]. Os dois *benchmarks* serão descritos a seguir:

1. Linpack

   Linpack ou também HPL (*High-Performance* Linpack) é um *benchmark* para resolver sistemas de equações lineares denso do tipo $A.x = b$, onde A é uma matriz densa gerada aleatoriamente de dimensão N x N e, x e b são vetores de tamanho N. Isso permite ao usuário dimensionar tamanho do problema e otimizar o software a fim de obter o melhor desempenho possível da máquina.

   Alguns fatores podem influenciar o desempenho de uma máquina, como: a arquitetura da máquina, o algoritmo utilizado para teste, o tamanho do problema, o nível de esforço humano para identificar as especificações da máquina, etc. As medições de desempenho realizadas pelo Linpack refletem a performance de um sistema dedicado para resolver um sistema de equações lineares densas. Segundo Whaley *et al.* [65] o desempenho alcançado é bastante elevado, se os testes forem realizados corretamente, podendo ser utilizado como uma boa aproximação do desempenho teórico de uma máquina.

   O Linpack representa o desempenho de uma máquina em *Floating-point Operations Per Second*, ou somente Flops. Flops é utilizado para determinar o desempenho de um computador no campo de cálculos científicos, ou seja, computadores que fazem grande uso de seus recursos para cálculos com ponto flutuante.

Para realizar a medição de desempenho, o *benchmark* utiliza bibliotecas matemáticas que proveem rotinas para as operações com matrizes como a MKL (*Intel Math Kernel Library*), a BLAS (*Basic Linear Algebra Subprograms*) e a ATLAS (*Automatically Tuned Linear Algebra Software*). A biblioteca MKL é implementação da Intel e é otimizada para processadores de sua arquitetura; a BLAS é uma biblioteca mais genérica para diferentes arquiteturas; e a ATLAS é utilizada para otimizar a BLAS para uma determinada arquitetura.

2. Bonnie++

   Bonnie++ [66] é um *benchmark* para mensurar operações de E/S (entrada e saída) de disco e avaliar sistemas de arquivos. Bonnie++ avalia tanto o trabalho realizado por segundo quanto a porcentagem de tempo que o trabalho levou sobre a utilização da CPU.

   Os testes realizados utilizando o Bonnie++ incluem velocidade de escrita e de leitura de dados, número de *seeks* por segundo, numéro máximo de arquivos escritos, lidos e/ou exluídos, etc. Com isso, este *benchmark* divide os testes de avaliação em duas partes. A primeira é a avaliação de *throughput* de E/S, e a segunda testa a criação, a exclusão e a leitura de arquivos.

As próximas subseções apresentam os testes planejados nesta seção.

5.2.2 Teste 1 - Provisionamento de VM

Para analisar o provisionamento de VM foram investigados fatores que podem influenciar o tempo de inicialização das máquinas virtuais. Para isso, foram levantadas duas hipóteses:

1. Primeiro foi levantada a hipótese de que recursos computacionais como tamanho de memória RAM ou número de núcleos poderiam influenciar o tempo de provisionamento;

2. A segunda hipótese é que o tamanho de disco rígido acrescentado nas VMs poderia também influenciar no tempo;

Para verificar se a primeira hipótese é válida foi analisado o tempo que as VMs, dos tipos pequena, média, grande e extra grande, levam para serem inicializadas na nuvem. Primeiramente, as VMs foram configuradas sem nenhum disco rígido e cada tipo de VM foi inicializada dez vezes para uma melhor aproximação do tempo médio de provisionamento. A Tabela 5.2 mostra o tempo médio e o desvio padrão obtido em segundos para cada tipo de VM.

Tabela 5.2: Média de Tempo e Desvio Padrão, em Segundos, do Tempo de Inicialização de Dez testes para cada tipo de VM.

Tipo de VM	**Tempo médio (s)**	**Desvio padrão (s)**
Pequena	6,320	0,951
Média	6,164	0,836
Grande	6,208	0,951
Extra grande	6,262	1,031

Se a primeira hipótese estivesse correta, as médias obtidas do tempo de inicialização deveriam variar bastante. Então, com os resultados obtidos a primeira hipótese pode ser descartada, pois mesmo inicializando VMs com diversas configurações, pode-se constatar que a média e o desvio padrão estão em uma mesma faixa de valores.

Como o tamanho de memória RAM ou a quantidade de núcleos não afetou o tempo de provisionamento de uma VM, para a segunda hipótese foi utilizado apenas o tipo de VM pequena, mas com diferentes tamanhos de disco: 10 GB, 20 GB, 40 GB e 80 GB. Foram realizados dez testes para cada tamanho de disco diferente. A Tabela 5.3 mostra a média e o desvio padrão do tempo obtido em segundos do provisionamento de VMs com apenas o tamanho de disco variando.

Tabela 5.3: Média e Desvio Padrão, em Segundos, do Tempo de Inicialização com Apenas o Tamanho de Disco Rígido Variando.

Tamanho de disco	**Tempo médio (s)**	**Desvio Padrão (s)**
10 GB	6,389	0,724
20 GB	6,139	0,791
40 GB	6,159	1,127
80 GB	6,330	0,950

Como a primeira hipótese, a segunda também foi descartada, pois mesmo com diferentes tamanhos de discos, a média e o desvio padrão se mantiveram no mesmo intervalo, inclusive o tempo de provisionamento de VM mostrou-se constante entre VMs com e sem disco rígidos.

5.2.3 Teste 2 - Escalonamento de Recursos

Um dos principais fatores para a escolha da plataforma CloudStack para a implantação de Nuvem de IaaS, como mostrado na Seção 3.3.5, foi a possibilidade do escalonamento horizontal de recurso para as máquinas virtuais.

Na nuvem implantada com o CloudStack existem três principais estados para as VMs: executando, parada e destruindo. O estado de executando é quando a VM está livre para uso ou está em uso, ou seja, está pronta para ser acessada. Quando se diz que a máquina

está no estado parada é quando a máquina não está acessível, entretanto, é quando pode-se realizar mudanças nas configurações (memória RAM, número de núcleos, disco rígido acoplado, etc.) de uma VM. Por fim, o estado de destruindo é um estado temporário, no qual a VM fica até sua destruição completa do sistema.

Neste contexto, o escalonamento horizontal só pode ser realizado quando a VM está no estado de parada. Além disso, o escalonamento horizontal está limitado pela capacidade do *host*, ou seja, os recursos que uma VM pode oferecer depende dos recursos que um *host* oferece. Entretanto, outro fator interessante que foi observado é que se um *host* não atende a capacidade requerida, há a possibilidade de migração da VM para outro *host* que tenha a capacidade requerida, desde que os *hosts* estejam no mesmo *cluster*. A Figura 5.5 ilustra o fluxograma com os processos de sucesso ou falha de escalonamento horizontal realizado na nuvem implantada.

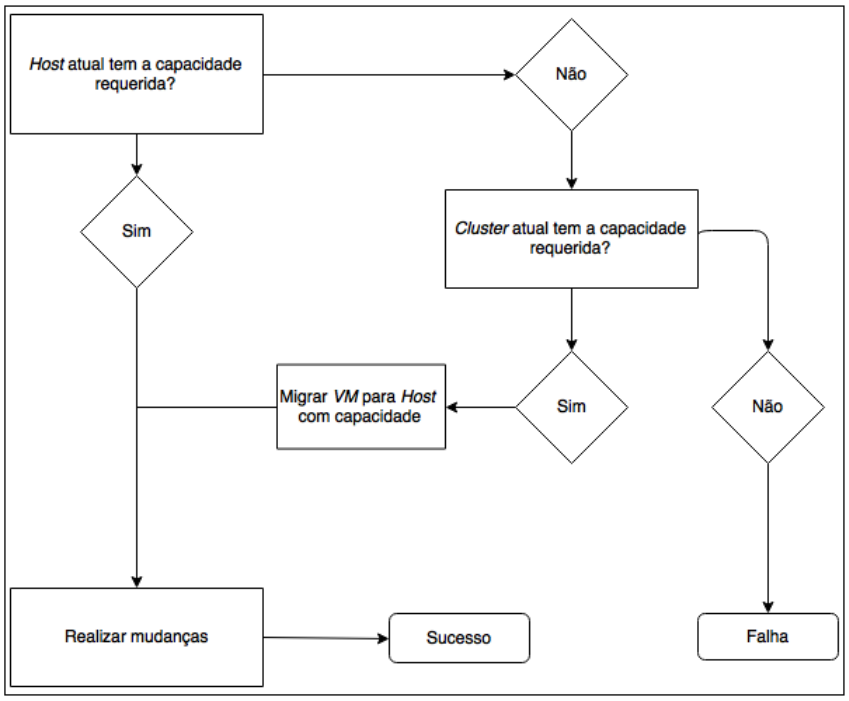

Figura 5.5: Fluxograma com o Processo de Escalonamento Vertical Provido pela Nuvem Implantada.

5.2.4 Teste 3 e 4 - Performance e Isolamento de VM

Para testar a performance e o isolamento de VM foi utilizado o *benchmark* Linpack com a biblioteca MKL, já que o *cluster* da nuvem implantada utiliza processadores Intel. Para o Linpack é sugerido que a escolha do tamanho N do problema seja escolhido para 90% da capacidade da memória RAM da máquina ser testada. Para calcular o tamanho

N, foi a utilizada da seguinte formula: $N = \sqrt{(RAM/8)}$. Por exemplo, uma máquina com 1 GB de memória RAM possui N igual a 10733.

Foram analisadas as performances dos quatros tipos de VM citados na Seção 5.2.1: pequena, média, grande e extra grande. A Tabela 5.4 apresenta os valores N utilizados para cada tipo de VM.

Tabela 5.4: Relação do Tamanho do Problema N Utilizado para Testes.

Tipo de VM	**Tamanho *N* do problema**
Pequena	10733
Média	15179
Grande	21466
Extra grande	26290

Para mensurar a performance foram realizados cinco testes para cada tipo de VM. Assim, foi possível obter uma média da melhor aproximação da performance de cada um. Os resultados obtidos giraram na ordem de GFlops (Giga Flops). Além da média foi realizado o cálculo do desvio padrão para demonstrar que não houve uma grande dispersão nos resultados obtidos. Além disso, os desempenhos obtidos foram comparados com o desempenho máximo teórico esperado de máquinas com as mesmas configurações de cada tipo de VM. O valor máximo teórico foi obtido utilizando a ferramenta HPL *Calculator*[6], que mostra valores de 70% a 100% de rendimento que a medição em GFlops pode se encaixar. A Tabela 5.5 mostra a média do desempenho de cada tipo de VM, o desvio padrão e o desempenho máximo teórico em GFlops.

Tabela 5.5: Relação entre a Média e o Desvio Padrão da Performance Testada com a Performance Máxima Teórica, em GFlops.

Tipo de VM	**Desempenho médio**	**Desvio padrão**	**Desempenho máximo**
Pequena	9,2030	0,0071	12
Média	18,4545	0,0049	24
Grande	18,9874	0,0011	24
Extra grande	18,9824	0,0046	24

A Figura 5.6 mostra a porcentagem da performance atingida pelas VMs em relação à performance máxima teórica, e a porcentagem máxima esperada com a utilização do *benchmark* Linpack. Os desempenhos obtidos giraram em torno de 76% a 79%, quando comparados com o desempenho máximo teórico. Vale ressaltar que os testes seguiram a recomendação de utilizar 90% da capacidade da máquina. Apesar da performance ter

[6] `http://hpl-calculator.sourceforge.net/`

ficado abaixo dos 90% esperados, os valores ainda se encaixam na faixa de 70% a 100% que a ferreamente HPL *Calculator* indica que que máquinas com as mesmas configurações podem estar. Além disso, os resultados mostraram que para cada tipo de VM o desempenho foi sempre constante, com pouca variação de seus valores.

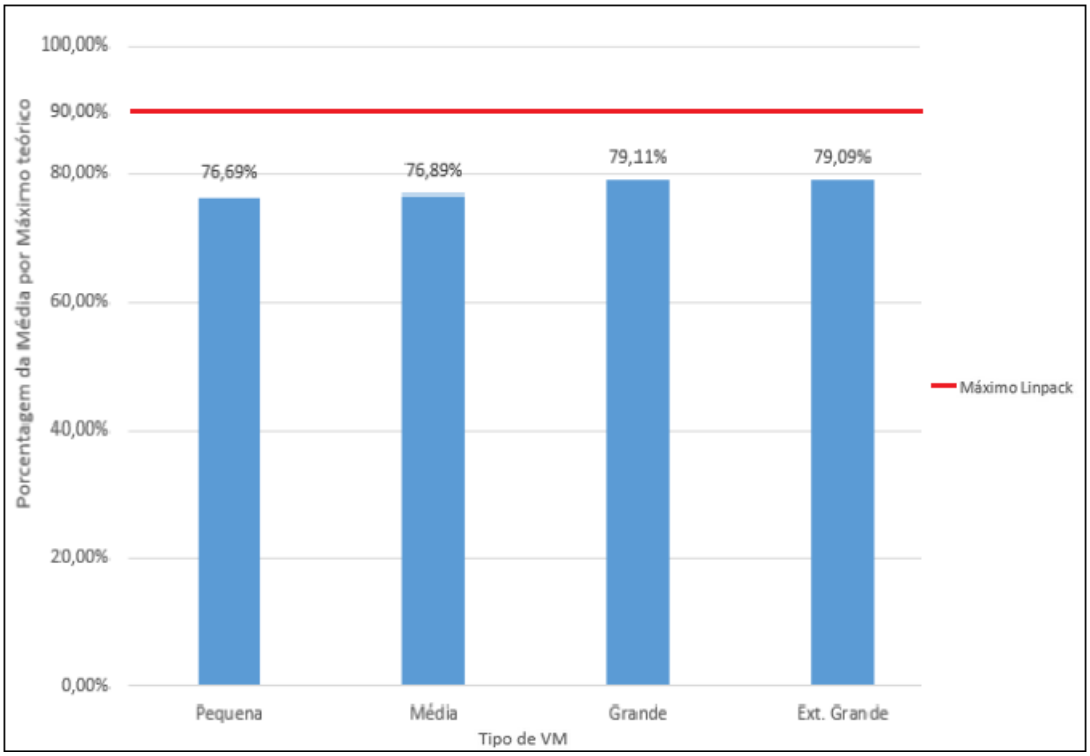

Figura 5.6: Porcentagem da Performance Média em Relação a Performance Máxima Teórica.

O próximo passo foi testar o isolamento das VMs. Então, para a realização do teste foi utilizada a seguinte estratégia: realizar a medição do desempenho de uma VM específica com o Linpack e observá-lo a medida que mais VMs são alocadas no mesmo *host* da nuvem privada implantada. Além disso, foi utilizada a aplicação Lookbusy[7] para simular um ambiente em que todas as outras máquinas tivessem sendo utilizadas. Foi atribuído o valor de 90% da capacidade total de processamento de cada máquina.

O teste de isolamento utilizou o tipo de VM pequena para que mais máquinas pudessem ser alocadas simultaneamente. A média e o desvio padrão, mostrados na Tabela 5.5, serviram como parâmetros para a performance de apenas uma VM alocada.

A Figura 5.7 mostra o desempenho médio obtido em GFlops do teste com a utilização de até seis VMs alocadas, simultaneamente, no mesmo *host*. Além disso, a área em verde na Figura 5.7 é delimitada pelo desvio padrão, de aproximadamente 0,007 Giga *Flops*, do teste de performance realizado com apenas uma VM alocada.

[7] `https://www.devin.com/lookbusy/`

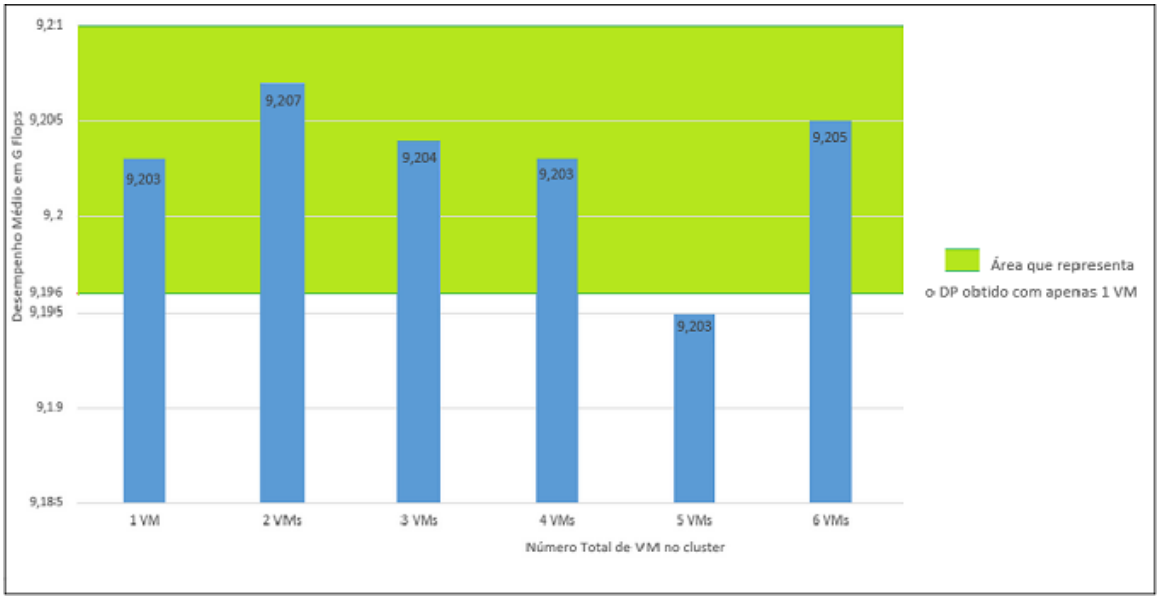

Figura 5.7: Desempenho Médio, em Giga *Flops*, Obtido com Diferentes Quantidades de VMs Alocadas na Nuvem.

As médias de desempenho, mesmo com várias VMs utilizando 90% de suas capacidades totais de processamento, ficaram dentro do esperado. O resultado com cinco VMs, alocadas ao mesmo tempo, apresentou um leve desvio dos valores esperados, mas, mesmo assim, a diferença foi muito pequena. Com isso, mesmo aumento o número de VMs simultaneamente alocadas as médias de performance ficaram dentro do desvio padrão de apenas uma VM alocada. Então, pôde-se considerar que a nuvem provê um ótimo isolamento de VMs, mantendo a mesma performance e desempenho, mesmo que várias maquinas estejam trabalhando ao mesmo tempo com quase a capacidade total.

5.2.5 Teste 5 - Operações de Disco

Para testar as operações de disco foi utilizado o *benchmark* Bonnie++. Para o teste foi utilizado dois tipos de máquinas, pequena e média, cada uma com um disco de 20 GB. O Bonnie++ permite escolher a quantidade de arquivos (criados pela própia ferramenta) para serem utilizados nos testes. Então, os testes foram realizados com 1024 arquivos, e como recomendado pela documentação do Bonnie++, os arquivos tinham o dobro de tamanho da quantidade de memória RAM, ou seja, arquivos de tamanho de 2 GB para a VM pequena e de 4 GB para a VM média.

A nuvem privada foi implantada com apenas um servidor NFS, ou seja, mesmo com o espaço de disco alocado separadamente para cada VM, elas compartilham o mesmo servidor NFS. Neste contexto, o teste de operação de disco visa analisar dois cenários de operações de E/S. Primeiro, foram testadas operações de escrita, de leitura e de exclusão de arquivos para cada VM separadamente. Depois, foi realizado o teste com duas máquinas simultaneamente.

A Figura 5.8 mostra a velocidade em KB/s para realizar as operações de E/S em duas séries. A primeira mostra o ambiente com apenas uma VM realizando operações de disco, e a segunda mostra o ambiente com a VM pequena e média realizando simultaneamente as operações de disco.

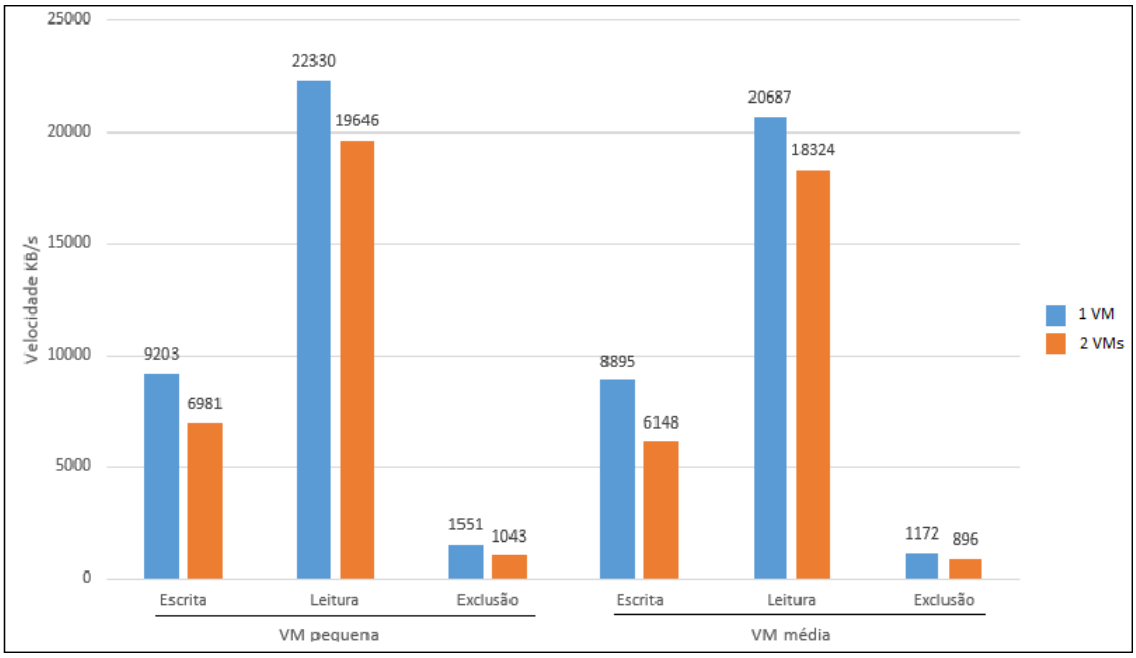

Figura 5.8: Velocidade em KB/s de Operações de E/S de Disco com NFS.

Pôde-se observar que em um cenário com mais de uma VM, o desempenho diminuiu em todas as operações. Rafael Avila [67] defende em sua tese que a característica de centralização do NFS pode gerar um gargalo se muitas operações forem requisitadas

simultaneamente. Assim, embora a velocidade das operações tenham diminuido, o resultado já era esperado, e o servidor NFS cumpriu perfeitamente sua tarefa de prover acesso compartilhado ao armazenamento em disco para as VMs da nuvem privada implantada.

5.3 Considerações Finais

Este capítulo apresentou os testes realizados na nuvem privada de IaaS implantada no LABID. Os resultados demostraram que a nuvem foi capaz de suportar a integração com a plataforma de Bioinformática BioNimbuZ, o qual foi um dos principais objetivos deste trabalho.

O teste realizado para analisar o comportamento da elasticidade vertical da nuvem, uma das características principais apresentadas pelo CloudStack, mostrou que além da possibilidade de aumentar os recursos de uma máquina virtual foi possível migrar VMs para outros *hosts* de um mesmo *cluster* da nuvem implantada para atender a demanda da elasticidade.

Por fim, os testes de performance demostraram que o provisionamento de máquinas virtuais possui tempo médio de inicialização com desvio padrão estável independente dos recursos computacionais ou tamanho de disco escalonados com as VMs. Além disso, os valores obtidos nos testes de performance e isolamento de VMs mostraram valores satisfatórios, e com isso, demostraram que a nuvem implantada fornece VMs com performance igualitária independente da quantidade de máquinas e/ou recursos requisitados. Com o teste de performance de disco, foi possível observar que existe um gargalo no sistema NFS utilizado para implantação da nuvem, pois a velocidade em kb/s de operações de E/S decai conforme a intensa utilização por diferentes VMs, mas todas as operações de E/S foram realizadas com sucesso.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho foi proposta a implantação e a avaliação de uma nuvem privada de infraestrutura como serviço para o LABID. O objetivo dessa implantação é colaborar com projetos do laboratório que utilizam nuvens de IaaS. Como, por exemplo, a plataforma de Bioinformática BioNimBuZ, que apesar de trabalhar com a federação de nuvens, não possui nenhum suporte de nuvem de IaaS privada dentro da UnB, e anteriormente a este trabalho todo o serviço dentro do escopo deste projeto era requirido para outras empresas e instituições.

Para isso, foi realizado um estudo e uma avaliação sobre as principais plataformas que oferecem a implantação de nuvem privada de IaaS. Para o contexto desde trabalho, a plataforma CloudStack foi a escolhida. A partir desta plataforma foi proposta e implantada uma arquitetura de nuvem privada utilizando os recursos disponíveis no LABID.

Com a nuvem privada já implantada, foram realizados testes para garantir que *workflows* da plataforma BioNimbuZ possam ser executadas no ambiente proposto. Para isso foram realizados testes de integração entre a nuvem privada e o BioNimbuZ, e testes para analisar o comportamento e o desempenho de aspectos chaves providos pela infraestrutura da nuvem implantada.

Como trabalhos futuros, sugere-se que novas plataformas possam ser utilizadas para comparar diversas implantações de nuvens de infraestrutura. Também, pode-se utilizar outros modelos de serviços, como PaaS e/ou SaaS para auxilar o ambiente acadêmico da UnB. Outro trabalho sugerido é abordar o tema de segurança para verificação de autenticação, autorização e confiabilidade de nuvens de IaaS.

Além disso, foi proposto um projeto de implantação de nuvem híbrida de IaaS para o Centro de Informática (CPD) da UnB. Assim, outros projetos, programas e atividades de ensino e pesquisa poderão compartilhar recursos computacionais, ou seja, estender os serviços de nuvem para a comunidade acadêmica da UnB.

Referências

[1] M. V. Steen and A. Tanenbaum. *Distributed Systems: Principles and Paradigms*. Prentice Hall Professional Technical Reference, 2 edition, 2006. x, 4, 6, 7

[2] I. Raicu I. Foster, Y. Zhao and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. IEEE, 2008. x, 7, 9

[3] L. Cheng Q. Zhang and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010. x, 11, 12

[4] A. Marinos and G. Briscoe. Community cloud computing. In *Cloud Computing*, pages 472–484. Springer, 2009. x, 13, 14

[5] J. Broberg e A. Goscinski R. Buyya. *Cloud computing: principles and paradigms*, volume 87. John Wiley & Sons, 2011. x, 17, 22

[6] Linux.com. The Top Open Source Cloud Projects of 2014. `https://www.linux.com/news/top-open-source-cloud-projects-2014`. Acessado online em 5 de abril de 2016. x, 23, 24

[7] OpenStack. `http://docs.openstack.org/`. Acessado online em 7 de abril de 2016. x, 23, 26

[8] CloudStack. `https://cloudstack.apache.org/`. Acessado online em 7 de abril de 2016. x, 3, 23, 28

[9] OpenNebula. OpenNebula. `http://opennebula.org/about/technology/`. Acessado online em 15 de março de 2016. x, 18, 19, 29

[10] Hewlett Packard Enterprise. HP Helion Eucalyptus. `http://www8.hp.com/br/pt/cloud/helion-eucalyptus.html`. Acessado online em 15 de março de 2016. x, 18, 19, 30

[11] Google. Search Engine Strategies Conference. `http://www.google.com/press/podium/ses2006.html`, 2006. Acessado online em 2 de fevereiro de 2016. 1

[12] P. Mell and T. Grance. SP 800-145. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011. 1, 8, 9, 11, 14

[13] Hugo Vasconcelos Saldanha. Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática. Master's thesis, Departamento de Ciência da Computação, Universidade de Brasília, 2013. 2, 35

[14] J. Dollimore G. Coulouris and T. Kindberg. *Distributed systems: concepts and design*. pearson education, 2005. 4

[15] H. Pourreza R. Eskicioglu P. Graham C. S. Yeo, R. Buyya and F. Sommers. Cluster computing: high-performance, high-availability, and high-throughput processing on a network of computers. In *Handbook of nature-inspired and innovative computing*, pages 521–551. Springer, 2006. 5

[16] R. Morrison. Cluster Computing: Architectures, Operating Systems, Parallel Processing and Programming Languages. *GNU General Public Licence*, 2003. 5

[17] C. Kesselman I. Foster and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance computing applications*, 15(3):200–222, 2001. 6

[18] S. M. Hashemi and A. K. Bardsiri. Cloud computing vs. grid computing. *ARPN Journal of Systems and Softwares*, 2(5):188–194, 2012. 7

[19] S. Venugopal J. Broberg R. Buyya, C. S. Yeo and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009. 8

[20] R. Grifth A. D. Joseph M. Armbrust, A. Fox and R. Katz. Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009, 2009. 9

[21] M. Rappa. The utility business model and the future of computing services. *IBM Systems Journal*, 43(1):32, 2004. 9

[22] I. M. Llorente B. Sotomayor, R. S. Montero and I. Foster. Virtual infrastructure management in private and hybrid clouds. *Internet computing, IEEE*, 13(5):14–22, 2009. 12

[23] S. Gopalakrishnan L. Coyne and J. Sing. *IBM Private, Public, and Hybrid Cloud Storage Solutions*. IBM Redbooks, 2014. 13

[24] Cezar Taurion. *Cloud Computing-Computação em Nuvem*. Brasport, 2009. 15

[25] D. Rodgers A. L. Santoni F. Martins A. V. Anderson S. M. Bennett A. Kägi F. H. Leung G. Neiger, R. Uhlig and Larry Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005. 17

[26] E. Hentges B. Thomé and D.Griebler. Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS. *Anais da 11a Escola Regional de Redes de Computadores*, 2013. 18

[27] Abiquo. Abiquo. http://www.abiquo.com/overview/. Acessado online em 15 de março de 2016. 18

[28] Apache. Apache VCL. https://vcl.apache.org/. Acessado online em 15 de março de 2016. 18

[29] Apache. Apache CloudStack. https://cloudstack.apache.org/. Acessado online em 15 de março de 2016. 18

[30] Convirture. ConVirt Enterprise Cloud. https://www.convirture.com/products_cloud.php. Acessado online em 15 de março de 2016. 18, 19

[31] Nimbus. Nimbus Infrastructure. http://www.nimbusproject.org/docs/2.10.1/summary.html. Acessado online em 15 de março de 2016. 18, 19

[32] OpenQRM Enterprise. OpenQRM. http://www.openqrm-enterprise.com/index-1.html. Acessado online em 15 de março de 2016. 18, 19

[33] OpenStack. OpenStack. https://www.openstack.org/software/. Acessado online em 15 de março de 2016. 18, 19

[34] Ubuntu. Ubuntu Cloud. http://www.ubuntu.com/cloud. Acessado online em 15 de março de 2016. 18, 19

[35] F. Wang G. V. Laszewski, J. Diaz and G. C. Fox. Comparison of multiple cloud frameworks. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 734–741. IEEE, 2012. 20, 21, 22

[36] D. Corcoran J. Morrison T. Lynn, G. Hunt and P. Healy. A Comparative Study of Current Open-Source Infrastructure as a Service Frameworks. *CLOSER 2015 - 6th International Conference on Cloud Computing and Services Science, At Lisbon, Portugal*. 20, 21, 22

[37] Q. Li Y. Gao X. Wen, G. Gu and X. Zhang. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461. IEEE, 2012. 20

[38] IEEE 802.1 Standard Working Group et al. IEEE standard for local and metropolitan area networks: media access control (MAC) bridges. *IEEE Std*, 802. 22

[39] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, RFC Editor, 1997. 22

[40] P. Mockapetris. Domain names-implementation and specification. RFC 1035, RFC Editor, 1987. 22

[41] L. Andersson and T. Madsen. Provider Provisioned Virtual Private Network VPN Terminology. RFC 4026, RFC Editor, 2005. 22

[42] S. Denazis J. Hadi Salim D. Meyer E. Haleplidis, K. Pentikousis and O. Koufopavlou. Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC 7426, RFC Editor, 2015. 22

[43] C. Sapuntzakis M. Chadalapaka J. Satran, K. Meth and E. Zeidner. Internet Small Computer Systems Interface (iSCSI). RFC 3720, RFC Editor, 2004. 22, 41

[44] Inc. Sun Microsystems. NFS: Network File System Protocol Specification. RFC 1094, RFC Editor, 1989. 22, 41

[45] J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511, RFC Editor, 2006. 22

[46] S. Farrell S. Boeyen R. Housley D. Cooper, S. Santesson and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor, 2008. 22, 26

[47] C. Mortimore B. Campbell and M. Jones. Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7522, RFC Editor, 2015. 22

[48] C. Wu Z. Zhang and D.W.L. Cheung. A survey on cloud interoperability: taxonomies, standards, and practice. *ACM SIGMETRICS Performance Evaluation Review*, 40(4):13–22, 2013. 23

[49] The New Stack. The New Stack. `http://thenewstack.io/about/`. Acessado online em 5 de abril de 2016. 23

[50] The Linux Foundation. CloudOpen North America. `http://events.linuxfoundation.org/events/cloudopen-north-america`. Acessado online em 5 de abril de 2016. 23

[51] OpenNebula. `http://opennebula.org/`. Acessado online em 7 de abril de 2016. 23

[52] HPE Helion Eucalyptus. `http://www8.hp.com/br/pt/cloud/helion-eucalyptus-overview.html`. Acessado online em 7 de abril de 2016. 23

[53] S. Hartman C. Neuman, T. Yu and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, RFC Editor, 2005. 26

[54] B. R. Moura and D. L. Bacelar. Política para armazenamento de arquivos no zoonimbus. 2013. 35

[55] H. H. P. M. Costa. *Controle de acesso na plataforma de nuvem federada BioNimbuZ*. PhD thesis, Universidade de Brasília, 2015. 35

[56] V. A. Ramos. *Um Sistema Gerenciador de Workows Cientícos Para a Plataforma de Nuvens Federadas BioNimbuZ*. PhD thesis, Universidade de Brasília, 2015. 35, 46

[57] J. Martin J. Burbank D. Mills, U. Delaware and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905, RFC Editor, 2010. 40

[58] P. Goyal P. Sarkar P. Radkov, L. Yin and P. J. Shenoy. A Performance Comparison of NFS and iSCSI for IP-Networked Storage. In *FAST*, pages 101–114, 2004. 41

[59] S. M. Mane M. Stephens J. C. Marioni, C. E. Mason and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517, 2008. 48

[60] M. D. Robinson and A. Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome biology*, 11(3):1, 2010. 48

[61] Universidade de Brasília. BioNimbuz. `https://github.com/bionimbuz/Bionimbuz`. Acessado online em 01 de agosto de 2016. 48

[62] Universidade de Brasília. BioNimbuz Client. `https://github.com/bionimbuz/BionimbuzClient`. Acessado online em 01 de agosto de 2016. 48

[63] L. Liu A. Paradowski and B. Yuan. Benchmarking the performance of openstack and cloudstack. In *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pages 405–412. IEEE, 2014. 49

[64] M. M. Al-Mukhtar and A. A. A. Mardan. Performance Evaluation of the CloudStack Private Cloud. *International Journal of Cloud Computing and Services Science*, 2(6):403, 2013. 49

[65] R. C. Whaley A. Petitet and J. Dongarra. HPL–a portable implementation of the high–performance Linpack benchmark for distributed–memory computers. *Available from Internet: http://www.netlib.org/benchmark/hpl*, 2016. 51

[66] R. Coker. Bonnie++. `http://www.coker.com.au/bonnie++/`. Acessado online em 25 de junho de 2016. 51, 52

[67] R. B. Avila. *Uma Proposta de distribuição do servidor de arquivos em clusters*. PhD thesis, PhD thesis, Universidade Federal do Rio Grande do Sul, 2005. 58